Fachbereich Informatik

Thesis (diploma)

Procurement of a Client Management System (CMS) for Haaland Internet Production

Steffen Nesper

Los Angeles, February 14th, 2003

Thesis (diploma)

To attain the academically grade of

Diplom-Informatiker (FH)

At the

Fachhochschule Konstanz

- Hochschule für Technik, Wirtschaft und Gestaltung -
Fachbereich Informatik/Wirtschaftsinformatik

(University of Applied Sciences, Constance)

Subject	Procurement of a Client Management System (CMS) for Haaland Internet Production
Candidate	Steffen Nesper Schillerstr. 14 73575 Leinzell steffen@nesper.de
Supporter	Prof. Dr. Dieter Schaal (FH) Ronald Haaland (HIP)
Submitted	Los Angeles, February 14th, 2003

CONTENTS

ABSTRACT

The target of this thesis is the introduction of a client management system (CMS) at Haaland Internet Productions (HiP), a web design and hosting company in Burbank, California, USA. The company needs a system to track orders and improve workflow.

HiP needs a system which not only tracks orders, but also stores all client information in a database. This client information can be used for a variety of marketing and contact reasons. It is an important and integral part of HiP's client relationship management (CRM).

The lack of a cohesive CMS at HiP caused many fundamental business problems, such as lost orders, missed billing statements, and over/under billing.

The research done during the investigation and analysis of the company and their needs should lead to a global system which totally fulfils the needs of HiP. This global system could be in the form of an off-the-shelf product with some customizations, or a completely new, in-house system. Either solution will have respective pros and cons; the goal is to reach a decision that best fits HiP's needs and situation.

The following is a concise version of the project. Particular emphasis is placed upon the single steps which made up the decision process, as well as the practiced techniques, methods, and their applications.

1 INTRODUCTION

The project and the thesis were done at the University of Applied Sciences, Constance (Fachhochschule Konstanz) in the area of computer science/economic analysis (Informatik / Wirtschaftsinformatik) in cooperation with Haaland Internet Productions in Burbank, California.

I would like to thank Dr. Dieter Schaal, Mr. Ronald Haaland and Mr. John Goodner for their absolute support, rich knowledge, and deep understanding. Their help was integral and instrumental in leading this project to success.

My very special thanks go to Mr. Roger Park and Mr. Brendan Maze for proofreading, and supporting me during my thesis.

Also I would like to thank Ms. Caroline Nuttall for being my first tester, debugger, and all-around problem-solver.

2 GENERALLY

This section describes HiP-Design and its background, what the exact task of the project will be, and what HiP expects of the project.

2.1 About Haaland Internet Productions

Haaland Internet Productions or HiP-Design (short: HiP) is a full-service Internet Provider which offers everything needed for a successful website and corporate presentation. The emphasis in on the unique: every site is custom-built to not resemble any other site. They are producing high quality sites, from small individual (or personal) pages up to worldwide-acting companies. Not only do the art work and the appearance of a website need to be good, HiP offers high standard HTML and dynamic developing to provide a strong basis for operation.

Another important part of the company is web hosting. While HiP usually hosts every site they develop, they also offer high performance web hosting for anyone, as well as application service providing (ASP) and database hosting. HiP has a lot of customers and hosts a couple hundreds of sites.

There are only a few full-time employees at HiP. The total headcount fluctuates due to freelance/project-based hiring system popular with computer programming, graphic art, and advertising businesses.

In 1998 two different companies, the web hosting and developing company of Ron Haaland and the design company of his sister, Lynne Haaland, merged to become HiP Design. From that point on, they became highly successful, defying the recent market collapse of web and internet companies, and stand in very good position today.

2.2 Setting of Tasks

The tasks during the thesis are the production or procurement of a client management system (CMS). HiP expects relief from daily handling of simple jobs to the complex planning of tasks and the co-ordination of projects over a period of time. This covers the whole scale of their business, from small static presentations to very large and dynamic projects with several hundred sites.

The system should cover the three major task fields of the company which are:
- Production (new customers and existing)
- Maintenance
- Hosting

In the first project meeting, HiP explained what they would expect from the system and showed it with an example of a common daily work process:

A customer has a problem or needs something done by HiP. He or she is calling an employee at the company. The employee makes notes on the call and the wishes of the customer. Every detail of the phone call should be posted in the database and stored for further references.

Things to be entered are, for example, the expected number of hours, a concrete problem description, and a completion date as well as eventual costs. It is also noted if the customer is a member of the Client Service Center, (CSC). CSC customers host their sites at HiP servers and have a contract with a certain number of free monthly updates.

The call also must be differentiated between the individual customers. If it is a new customer, then his contact information is put into the database. (This is not a common situation because clients normally come to HiP and negotiate services and contracts first, and then only later call the office and request needed services.)
In the case of an existing customer, there must also be the possibility of adding them to the database, because it could be the first time the client calls, or it might

be a recently acquired account. If the client is a member of CSC must also be viewable, as well as the remaining free updates in their account.

After the information is entered into the system, the goal is to keep track of everything that will be done for a client, such as phone calls, uploads, programming, image processing, design and so on. The system will show the actual state of the task; after it is done, it needs to be checked by a supervisor and then sent to the billing department. Billing is not a part of the system: There is already a working environment in use. The system should be a total information system for the employees of HiP as well as the management.
An employee could look up all necessary client information as well as their own worked hours, or gain information about ongoing projects if he or she needs to start working on a project mid-stream. The employees should have somewhere they can keep track of their open projects and tasks. It is obvious that there is a need of different access levels throughout the whole system.

2.3 Summary of the required functions

In conclusion to the above stated facts, the system needs to fulfill the following tasks:

- Order tracking and project management for the three major fields
- Ability to extend the system if there are new fields in which HiP will be active in the future
- Web design and programming of new sites
- Maintenance of web pages from client service center customers and one-time projects
- Web site hosting with email and ftp setup
- Information system for clients and client contacts, contracts and ongoing projects
- Tracking of employee hours per week/month
- Task list per employee with the ability to change the priority and send reminders
- Different access levels for employees (roles)

2.4 Goal of the project

The Goal of the project is the introduction of a working system which fulfils the needs of HiP to track every single task and also find out if there are possible improvements in the daily workflow. First there will be an analysis of the daily work, followed by discussions with employees of HiP and some of their customers to get suggestions on how the system should be and what features they need. Then there will be a meeting in which the project team delineates the requirements and wishes of a Client Management System. After that there will be another analysis and information about existing systems which could be used or adapted to the needs of HiP.

The management of HiP would like to develop their own system because they might create something usable for the new projects of their customers. It would also match the needs of HiP much better, and be much more satisfying for the project team, if they develop something of their own. After the analysis phase there must be a decision: either for an off-the-shelf/adapted system, or for an in-house developed system.

A test system should be available by the third week of October, with a finalized and usable version to be running by December, 31st 2002.

3 INVESTIGATION OF THE CURRENT SITUATION AT HIP AND THE MARKET

Making a decision in which direction the project will go depends upon a wide investigation of the software market and the current situation at HiP.

3.1 Methods to analyze the environment

Some special techniques and methods were used to make a comprehensive decision.

3.1.1 Requirement analysis at HiP and some clients

The first step was to understand the needs of HiP and its employees. For that purpose, there was an initial meeting to talk about the CMS and its features. Every employee made some suggestions and participated in brain-storming sessions. The project leader then put everything together and made a rough questionnaire with which he interviewed employees and two clients: What do they expect of such a system, what features would they like it to have, and would they be willing to use it? In another meeting, he presented the gathered information and they lined out the requirements in detail.

With this information in hand, the analysis of the current workflow began. It was basically the same at everybody's workflow. A client calls somebody in the company and the employee writes down the requests. Some did it in a computer document, others did it on paper. Sometimes it was a task for another employee and they passed the information on to a supervisor. After the job was done, the employees usually reported completion to the billing department and/or to a supervisor for examination. After receiving approval from a supervisor, the billing department billed the client for the work.

It was almost inevitable that something, somewhere, would be misplaced or lost, and indeed it sometimes happened. If nobody worked on the task, the client usu-

ally called again awhile later, wondering what was going on with his or her project. This was damaging to the company's reputation as a top-level web development and hosting company. The other, and much more important issue, was when an employee performed the work but forgot to inform the billing department. The company would therefore not be able to collect valuable billable hours, a main source of income. HiP also wanted to keep track of all hours spent on all tasks over a week, month, or other span of time. In essence, the most important thing would be a system to manage and keep track of all tasks and jobs at HiP.

There was a wide range of needs and demands from the clients. One client said that they would also like to be able to keep track of the project, i.e., to log on and check the current status, see who is working on it, and perhaps upload more information after the order was placed (if there are any add-ons necessary). Another client preferred a "personal touch," wanting to talk directly to employees and not willing to use such an advanced system. Another was not interested in the particulars of progress at all.

Summary:

The analysis of client and employee needs revealed a lot of functions that needed to be built into the system. Some desires were mentioned by all, some only by individual employees or clients. Other functions address other features that do not appear to be necessary or obvious, but would be helpful for operators of the system, and improve the functioning of the client management system.

Obvious required business and client management functions are probably implemented in all pre-existing CM systems that HiP would consider purchasing. But an important feature missing from these existing CM systems are the viewable functions that may or may not need to be displayed or filtered. In the end, HiP decided upon a list with ideas and functions that needed to be part of the system, or could be there to make operations easier.

The mandatory functions are:

- tracking of tasks
- storing client information

- employee hours
- easy and fast to use
- offer support for Windows and MacOS or run on a web interface.

Optionally there could be:

- information pages for clients
- listing of frequently asked questions
- hour estimation on new projects
- task lists for employees
- reminders for tasks and sign-up/cancellation notices
- change user passwords for email and ftp.

3.1.2 Information about existing products

A week of research into existing software solutions revealed very few products which could satisfy the needs of HiP. The listed functions in 3.1.1 are often highly specific and include have a wide variety of data. It seems to be much more complicated to purchase and modify a system that fulfills those tasks. The project team was down to three products which could only partially fulfill some of the requirements of HiP.

3.1.2.1 DKAT Client Management Database (CMD)

(Information taken from their website)

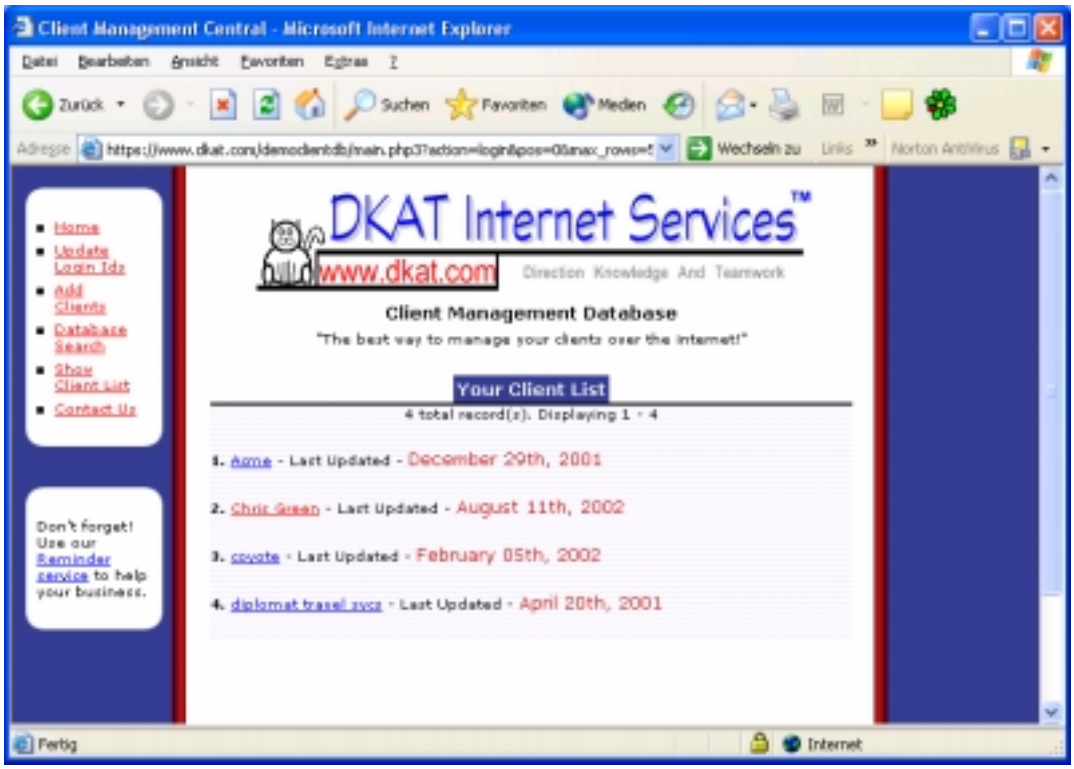


Fig. 1: ClientManagementDatabase, Screenshot[1]

"Our securely encrypted client database system was created to maintain up-to-the-second information about your client base.

With our online client management systems, you can instantly view the latest client information logged by your reps and sales agents. You can search your database of client information to find just what you need. Your employee can keep a client's information updated from anywhere as long as they have internet access. You can prevent an employee from leaving the company and taking their contact information with them. You can give a restricted password to each employee so that they only add information to their assigned client records. We can customize our system to meet your needs."[2]

[1] Taken from www.DKAT.com/db

[2] compare @DKAT

3.1.2.2 Optigold ISP

(Information taken from their website)

Fig. 2: Optigold ISP, Screenshot[1]

"Realtime Credit Card Processing

Realtime credit card processing is a beautiful thing (no, importing and exporting of transaction files is not what we're talking about when we say realtime). Let your billing system talk directly to your merchant account over TCP/IP and process transactions in 1-2 seconds. Approved transactions are automatically posted and applied to invoices. A declined credit card can automatically trigger a notice to the customer. (Optigold supports non-realtime processors/systems if you would prefer to go that route).

Sales Tax

Optigold ISP has the ability to charge sales tax if needed. Specify sales tax on a customer and product level. You can even specify sales tax classes for different sale tax rates. International and secondary taxes are supported. Special 'Texas Taxes' are also fully supported.

Customer Interface

You can give your customers the ability to check their accounts online. They can view things like payment history, invoice history and usage history. Customers can see what date they have paid for service up through, what the balance on their account is, etc. If you would like, you can give them the ability to manage their

[1] taken from http://www.digitalpoint.com/products/isp

own email addresses (adding, deleting, changing passwords). You can even let customers go online and make a one-time credit card payment in real time (maybe they are a cash/check customer that forgot to send in their payment this month). A survey can be taken by the customer that allows you to compile the results into handy marketing reports/charts. Or maybe you just want to let your customers edit their address info or credit card info as needed. All of it can be done with Optigold ISP.

Online Signup

Give yourself the ability to signup customers online. The online signup portion of Optigold ISP can do as much (or as little) of the signup process as you wish. Collect customer info, invoice customer, charge credit card and set new user up on system server(s) can all be done automatically. The customer can be completely ready to go with the services they selected (and paid for) without any employee intervention.

The online signup also has complete support for the most popular CD signup systems out there (Gearbox, IEAK, Netsurfer, Total Internet, Connection Magic and ISP Register).

Tech Support

If your tech support personnel are in a remote location (or even if they are not), you can give them the ability to access and log support issues via the web interface. They can optionally view the customer's emails (with passwords), domains, broadband configuration, etc.

Resellers/Salespeople

Your sales force can log into their web interface to add or view their existing customers. You can optionally give them the ability to view their commissions (history, due and pending). If you want to give them added control, you can allow them to use the ISP Management web interface designed for virtual ISPs and outsourced billing.

Multiple Protocols

Optigold ISP can control and provision your servers through a wide array of different protocols. Communication with your servers can happen over SQL/ODBC, JDBC, LDAP, telnet, FTP, RSH, SSH, HTTP (GET or POST), XML, SMTP/Email or a direct command line. You can even have Optigold ISP dial a phone number as a server event (maybe you running a paging service and you want to page your customer when they go past due).

What that means is that virtually anything you can do as an administrator, Optigold ISP can do for you.

Fig. 3: Optigold ISP, servermodule[1]

Triggers

You can define what triggers you want to use (for creating accounts, deleting, placing on hold, changing password, provisioning domains, emails, aliases, etc.) You can even define non-specific generic actions for whatever special functions you can think of.

You have the ability to make your server events triggers global or you can make them specific to individual products/services you sell, which gives you the ultimate flexibility to provision anything completely based on what the customer specifically purchased.

[1] taken from http://www.digitalpoint.com/products/isp/server.html

Server events can call an infinite number of actions per trigger, across all supported protocols. Server actions can also be specific to a virtual Internet Service Provider (ISP) for multi-ISP setups."[1]

3.1.2.3 Magic Service Desk Enterprise Editon

(Information taken from their website)

"Complete Management Toolkit

Provides all the tools necessary to manage enterprise support proactively, including a fully functional desktop management suite, remote control capabilities, and robust event and network management solutions.

Retirement of List Values

This feature-added to a variety of tables in Magic, such as Subject, Client, Group and Staff-allows items to be tagged as "Inactive." Inactive items will not appear in any lists, nor can they be selected. However, these values are still available for query and reporting purposes.

Bulk Update of Tickets

This feature on the Tech Monitor and Helpdesk Monitor provides the ability to select multiple records, such as Tickets, Work Orders, PRs and other files. The user can then assign, close, and reopen the set without having to open each record individually.

Full Keyboard Support

In 2001, the U.S. Government issued a mandate regarding software usability for the visually handicapped, known as "Section 508." Magic Enterprise Edition is completely "508" compliant and now allows access to all of its main functions through the keyboard, without requiring mouse controls.

[1] taken from http://www.digitalpoint.com/products/isp

Simultaneously Search Tickets, Knowledge Packs and External Documents

The original External Document searching function required the user to scan tickets separately from external documents. This release of Magic allows the user to canvas all sources at one time, or selectively pick the source to limit their search.

Search by Incident Number Quick Find

Users can now search by incident number without having to click on the "Find All" button. From the incident form, the user simply types an incident number into the "Search By" field and Magic will pull up the appropriate incident. This also works for other modules using a unique ID, such as work order number or purchase request number).

First Call Resolution (FCR)

FCR lets users mark incidents that are resolved on the first call. It includes statistical reports showing the number and percentage of calls closed on first call.

Subject Tree

In form customization, administrators can now choose the method of displaying subjects. For example, they can use the current Subject Popup view or the 6.x style Subject Tree view.

Automatic Client Comment Popup

When looking up a client in the Incident or Incident Monitor screen, administrators can request that Client Comments automatically pop up. This is helpful for notifying technicians that a particular client is an executive or has special needs."[1]

[1] taken from http://magicsolutions.com/products/help/default.asp

3.1.3 Test of systems

HiP tested to see if the chosen off-the-shelf software could be used. Firstly, the overall appearance of the systems was appraised and, later on, a test of the special functions required by HiP.

3.1.3.1 DKAT Client Management Database

There is a nice online demo of the DKAT CMD available on their webpage. It looks really simple and easy to learn. But first impressions can be deceiving: it does not have many functions at all. It is only a tool to manage client contacts. It is written in PHP, however, so it would be possible to add features to match the needs of HiP. The layout and appearance could be totally customized, and the customer support is really nice and helpful.

3.1.3.2 Optigold ISP

Optigold ISP makes from the first second a really nice impression, and it looks like a professional tool. The installation is simple and easy. There are demo versions to download available for Mac, Windows and Linux online (we tested only the Windows version). The major drawbacks are that it is limited to 100 clients and there is no web interface in the demo version. It is also a stand-alone program which can not connect to an external database, rendering it workable on only one machine. The client and user management is really complex and offers a lot of features. Clients are able to log in to the system and setup email and manage their accounts. The billing module seems to be complete but is not a feature HiP wants to use, as they have an existing billing system they like very much.

There is also an excellent server management tool. This was not necessarily a feature HiP wanted, but after trying it out, they enjoyed its performance and decided that it would help do things faster in server management. There is also a good frequently asked questions (FAQ) section in which the Help desk adds commonly asked questions about problems and posts it online so they can refer the next time to that page. It will also print out statistics of those FAQ's, which could then be

used to determine problem areas with customer support. The order tracking of the system is highly specific because the program was built for an internet service provider. Optigold ISP would be perfect for the hosting of clients, but for developing and designing of websites it would be necessary to add some features. Digitalpoint, the makers of Optigold ISP, do offer that service.

3.1.3.3. Magic Service Desk

The feature list of Magic Service Desk seems to be complete and could match the needs of HiP. Unfortunately, the ordered demo version never arrived at the HiP offices. After asking what happened, they promised to send out a copy immediately but it also never arrived. The project team decided that any company that had such poor customer relations probably had a lot of other problems as well, and as such, Magic Service Desk was ruled out.

3.1.4 Matrix analysis / Comparison requirements and features of system

	Optigold ISP	DKAT CMD	Magic Service Desk*
price	$1,000 + $1 per customer	$ 569	
	Database Server: $215		
	Web Server: $429		
	$64 per employee		
price for 500 customers and 10 employees	$2,784	$ 569	
functions			
order tracking	only for ISP	X	
client management	YES	YES	
employee hours	X	X	
easy, not too many functions	X	YES	
Web interface or Mac and Windows	YES, Web, Mac, Windows and Linux	YES, Web	
outside information for clients		X	
FAQ	YES	X	
hour estimation	X	X	
task list for employees	X	X	
task reminder	X	X	
change passwords	YES	X	
billing	YES	X	
options			
Server management	YES	X	
upgrades	free	?	
source code available	X	YES	
* not tested			

Fig. 4: Matrix of the tested features and results

The above matrix (Fig. 4) shows all features that should be a part of a CMS which fulfils the needs of HiP. After extensive testing and research, the gained results were discussed in a project meeting.

3.2 Discussion and results of the analysis and test

After the HiP environment was decided, the project team analyzed the results and weighed the choices.

3.2.1 Results of analysis

The extensive testing and installation phase of the various products opened up some new solutions; the project team had differencing opinions on how they should act and deal with the gained results. The first product, Magic Service Desk, was not considered anymore because they could not deliver a demo version of their product in time.

DKAT CMD is a nice and cheap product and has the ability to add the functions HiP needs. The basic functions are not enough, however. The program only offers the ability to manage clients and client contacts. It is based on PHP and none of the developers at HiP are really familiar with that language, so an extension of the program would be complicated. The basic functions of DKAT CMD could be built in a few days and probably be cheaper (the view of most project team members) they decided not to go with DKAT CMD.

Optigold ISP looked like the most complex and best product in the test range. It offered almost every function HiP needed and a lot more which could be needed in the future. Some of the functionality is complicated and needs trained employees to get it working right. The source code is not available and the extensions considered necessary need to be done by the software developer. The price of the product is not that high, so if the cost of customizing the order tracking module would also remain low, this might be the product to purchase.

3.2.2 Advantages and disadvantages make or buy

“People often ask if they should buy an off-the-shelf system or a system that can be tailored to meet their particular needs. They have generally heard about the comparative advantages and disadvantages from system vendors. The sales arguments generally run along the following lines:

Off-the-shelf systems can be purchased and implemented quickly. The systems are mass-produced and aimed at the mass market, so they are relatively cheap and they (usually) work. They are used by many companies. The disadvantage of an off-the-shelf system is that it may not match the company's needs. It may not be powerful enough to handle all the company's data, it may not have all the functionality that is required, or it may not allow the company to carry out its processes exactly as it did in the past."[1]

An own developed system fits perfectly to a particular company's needs. It handles the types and volumes of data the company uses. It has precisely the functionality the company needs. Its workflow functions correspond exactly to the way the company wants to work. The disadvantage of a custom made and unique system is that it is built expressly to the company's requirements. This takes time, so the system is not available immediately. It also requires development effort, so the system is not as cheap as an off-the-shelf system.

3.2.3 Decision over making an in-house system or buying an off-the-shelf product

After discussing the results and the advantages and disadvantages of making the system in-house or buying an off-the-shelf product, there were some other points to consider. The only usable off-the-shelf version, Optigold ISP, is made for an internet service provider and only represents some basic functions which were part of the original requirements analysis of HiP. The only function which is not part of the software is an important one, and Digitalpoint, the producer of Optigold ISP, is willing to add the function. They could not, however, fix a date for the completion of the modification. They only gave HiP quotes on their estimated costs to develop the changes, which often means that the customizing of the product extends the initial cost by a factor of two or three. A lot of functions in the software are really nice but HiP doesn't need or ever use them (like billing). So the advantage of "immediate availability of an off-the-shelf product" disappears. HiP needs to wait till the implementation of the extra features is done. The only

[1] taken from http://www.johnstark.com/pd63.html

advantage of the system would be that if something goes wrong or it doesn't work HiP could call customer service and get it fixed.

On the other hand, if they are producing their own system, it will take some time and a lot of manpower to implement it and match the needs. It will be necessary to test and make sure it will work properly under all conditions. If there is a problem, someone outside the company needs to take care of it. But this product will match the company needs 100% and if there are functions needed in the future they could be easily added. The appearance and the navigation would confirm to Corporate Identity. It seems to be also a lot cheaper than the customized off-the-shelf version of Optigold ISP.

The best way for HiP is to make their own system. This means HiP will have a system that fulfils the company needs by 100%; only the needed functions will be used by the company. There would not be any major problems developing in such a project because HiP has a lot of experience in that field and the most of the system will be basic in- and output as on every dynamic webpage. A lot of projects were done by HiP in the past, which showed that they have the competence to develop such a system with much more complexity and a lot more people involved.

4 DEVELOPMENT OF A CLIENT MANAGEMENT SYSTEM

After the decision was made to develop an own CMS there were a lot of methods and techniques used. To make the development a success extensive planning was necessary. The after analysis process will be shown in this chapter.

4.1 Project management

Project Management is vital to the successful completion of any project. For every project done by more than one person there should be clear roles about responsibilities and the setting of milestones. Even in small one-person-projects there should be at least a rough layout and some important milestones.

4.1.1 Responsibility

It is obvious that there is the need of clear roles and responsibilities in the project team. The project team consists of the following people:

Steffen Nesper - project leader
Ron Haaland - Chief Technical Officer HiP (CTO)
Lynne Haaland - Chief Executive Officer HiP (CEO)
John Goodner – visionary and project team member
Caroline Nuttall - project team member

The tasks of the project leader will be the whole management of the project and the developing of a working Client Management System. He will create a milestone plan, plan the whole project and lead it to a successful conclusion. After the project team has decided to make an own system the project leader will probably develop the most of it. The other team members will only be there to assist and help guide the project on the right path. They will attend all meetings and discuss their concepts with the project leader. In the project meetings they will talk about

current issues and decide the open points. In the end, they will join the product presentation and afterwards take the system into production status.

4.1.2 Milestones

Without milestones there is a chance that a project will not be completed in time. It is important for the success to set milestones and reach them. The original milestone plan only showed the first part of the system with its analysis of the current situation and the decision making as well as the mandatory end date. After that it was not clear in the beginning how the project would go on. After September 5th, the day of the decision that the project team will develop their own system, the milestone plan was changed and the points of developing were added to it.

The milestones are (all in 2002):
08/01 - start of project
08/15 - requirement analysis
08/26 - getting information about existing products and test them
09/05 - decision; make or buy
09/09 - final use cases done
09/11 - wire framing done
09/18 - application architecting done
10/01 - fuse coding done
10/14 - application integration done
10/15 - launch of first version
12/02 - implementation of all major functions done
12/23 - system tests done
12/31 - final version and launch of system

There is a detailed milestone plan available in the appendix 12, page A7.

4.2 Technical realization of the development

To make a project a success it is important to note what kind of technical features and systems are involved in the development process. It is also vital to know what technology it will be based on after it is done.

4.2.1 Technical possibilities at HiP

HiP is using a lot of different systems and is willing to buy new or extend existing equipment to fulfill the needs for a functional CMS. However, to keep it simple and offer the highest standard for their customers, one of the goals is only using reliable systems and be specialized on only some of them instead of knowing a lot of systems. So, it would be highly appreciated to cut back on the variety and concentrate on the few working systems and software applications.

4.2.1.1 Programming languages

The programming language used at HiP is mainly ColdFusion (CF) (see also paragraph 4.2.5) and HTML. The fact that it will be a dynamic system leads to ColdFusion. They set up also PHP but they don't produce anything with it. Microsoft's ASP, Java based JSP or other technologies are not yet used at HiP. There is also a good knowledge in shell scripting and C++ programming but for fast web application developing and the widest support of all developers ColdFusion is the recommended language by HiP for the project.

4.2.1.2 Databases

In use are Microsoft SQL, PostgreSQL and MySQL databases. The MS SQL runs only on Windows based boxes, PostgreSQL and MySQL are available for Windows, Linux and UNIX (and others). PostgreSQL was used a lot in the past but didn't fulfill the needs of the company and it will not be used on further projects, MS SQL needs a powerful machine based on Windows. There are a couple strong database servers available in the server farm but more of them are running the Linux operating system. The Linux based databases instead are also running on a

less powerful machine and seems to be more stable then the Windows versions. The log files and the experience of the past have shown that a Linux server needs much less maintenance than the same system running in a Windows environment.

4.2.1.3 Server environment

HiP has a couple of web and database servers. The most of them are running Linux as an operation system. Windows was used a lot in the past but the new machines are based on Linux. There are a couple reasons for that decision. Linux is free and the servers run usually more stable and offer a lot of more remote control then the windows based servers. HiP started using Apple Macintosh servers a while ago as well. They run stable and are also powerful. They thought about using more Macs in the future since there is a new operating system available. It is called MacOS X and is based on FreeBSD. The system runs stable and almost every application for Linux/UNIX can be ported on that operating system. It can be controlled remotely via SSH like the Linux servers. But after just a couple months on the market it hasn't shown his reliability jet but it will be in the mind for further servers, but for now they will stick with the proven Linux.

4.2.1.4 Performance

The performance of the Client Management System is not really important since there are only a few users using it at a time. Therefore, it basically depends on the used technology which type of machine it needs for a stable and problem-free operation.

4.2.1.5 Security

The security of the system is a very important part since there is a lot of internal information stored in the system like client contact, discussions about the clients' needs and so on. On the other side, it should be available over the internet because some employees are working from home or other places in the world and they should always be able to access the data in real time. So there will be always a

compromise between security and availability. The best solution in the middle is to fulfill all the needs should be taken.

4.2.2 Summary of the technical facts

Since there are only programmers who are using ColdFusion for 99% of their daily programming language, there is only a little or no knowledge of other languages like PHP. ColdFusion is therefore the realistic choice as the main developing platform. The project team is willing to use other techniques and technologies but after the project is finished. One of HiPs goals is cutting back on Microsoft servers in the future because of the license politics and the higher costs for hardware as well as maintenance. They would be grateful if there will be only Linux based products. ColdFusion runs on both Windows and Linux but is used at HiP through Linux. They are not looking into Mac because it is new and hasn't proven its reliability. As they don't want to use PostgreSQL anymore, the only usable database will be MySQL. MySQL has proven to be a reliable and stable database and it is free. It is an open source project which will be maintained and developed in an ongoing progress.

For security there will be a global login and only people with the appropriate access rights can go into the system. Secure data like passwords will be transferred via secure sockets layer (SSL) and stored encrypted into the database. Probably for higher security there might be the use of a secondary database only for password storage and the users will be stored on a separate database and they will be connected only via keys.

4.2.3 Pros and cons of the gained facts

The only disadvantage about using this system is the fact that ColdFusion was bought by Macromedia a while ago and it is not completely clear what future the product will have. MySQL is an open source based database and hasn't implemented some functions of the more professional database MS SQL. However, these are minor problems and the features can be implemented in a different way to meet the needs as they unfold.

4.2.4 HiP is going to use ColdFusion as the main language as well as MySQL

In the summary of the above facts, HiP will use ColdFusion and MySQL on a Linux based server. Since all products are up and running and there is no need of any new hard- and software; the installation costs for the product are not measurable. It is also highly recommended by the management of HiP that those technologies are used for the project. For the application and the database there will be two different servers. All Licenses are valid and there is no problem on the technical side to make the implementation a success.

4.2.5 What is ColdFusion

"Cold Fusion is a web language that combines an extraordinarily approachable and productive scripting environment, effortless connectivity to enterprise data, and a powerful suite of built-in application services. These features help one to rapidly build and deploy dynamic content publishing systems, self-service applications, commerce sites, and more. It is easy to learn and understand. Every programmer or web developer who has some experiences about developing websites will be familiar with Cold Fusion in no time. After a while they will develop applications in record time. It is an intuitive tag-based language that requires fewer lines of code by handling low-level programming tasks automatically and simplifying code reuse. The new server-side ActionScript[1] that enables Macromedia Flash developers to use the same scripting language for both client and server logic also improves the floss of developing websites in short times. There is a complete support for new ColdFusion features within the Dreamweaver MX development environment, including powerful visual layout and prototyping, enhanced code editing and development capabilities, and integrated debugging.

It is powerful and delivers a more compelling user experience while providing a robust architecture and rich set of built-in capabilities to deliver high performance and scalability.

[1] ActionScript, compare @ACSC

ColdFusion also offers fully integrated application services for adding full-text search, dynamic charting, and high-performance connectivity to Macromedia Flash clients to your applications. The innovative architecture that delivers the scalability, reliability, and power of the Java platform without the complexity is also a feature of ColdFusion as well as the complete extensibility via custom tag libraries, reusable components, Java/C++, and thousands of available third-party add-ons. Cold Fusion MX[1] supports open industry standards and easily integrates with your existing technology infrastructure. Cold Fusion is highly approachable integration with all of the major Internet standards and component models, including XML, web services, Java, .NET/COM, and CORBA. It also has support for developing and deploying applications on a standalone Cold Fusion server or on leading Java application servers such as IBM WebSphere Application Server. ColdFusion offers support for the leading server operating systems, web server software, mail servers, directory servers, file systems, and relational database management systems."[2]

4.2.5.1 Fusebox

"Fusebox is a framework consisting of a set of helper files and organizational principles, and a methodology consisting of a set of best practices for managing web projects. Used by application designers, developers and architects, the system addresses development problems such as unmanageable complexity, wasteful redundancy of effort, time-consuming code maintenance, and slow development speed.

With Fusebox 3.0, a clear separation is made between the framework and the methodology. The methodology associated with Fusebox is called the Fusebox Lifecycle Process (FLiP).

[1] ColdFusion MX is the actual version of ColdFusion (after CF 5)

[2] compare http://www.macromedia.com/software/coldfusion/productinfo

The Fusebox Framework

The first part of Fusebox is the "framework". The Fusebox framework consists of a series of files that aid in the creation of a complex web application. These files, known as the "core files" allow you to easily achieve nested layouts, inherited variables and settings, and break your application into more manageable pieces. Additionally, a set of best practices for organizing applications falls under the framework.

The goal of the Fusebox framework is to create a standard process for architecting web applications. This goal extends beyond a single technology. The core files have been duplicated for multiple web application languages.

The Fusebox Methodology

The second part of Fusebox is the methodology. The Fusebox methodology known as the Fusebox Lifecycle Process (FLiP) introduces an elegant new way to managing the software development process. The goal of FLiP is to reduce the atrocious 70% software failure rate. Through a very specific process, FLiP helps developers by providing them a series of steps for developing a web application: Everything from when and how to talk with your clients down to architecting and coding the application using the Fusebox framework."[1]

4.2.5.2 Fusebox Lifecycle Process

"The Fusebox Lifecycle Process is a process for developing web applications. FLiP grew out of some of the best practices employed by Hal Helms[2] and other members of the Fusebox community in the early days of Fusebox. Although, as the name indicates, FLiP came from the Fusebox community, its use is not intrinsically part of a Fusebox application. In fact, the ideas found in FLiP may be successfully employed in non-Fusebox projects as well.

[1] taken from http://www.fusebox.org

[2] Hal Helmes, one of the founders of Fusebox.org

The fundamental idea behind FLiP is to use a process that is at all times closely tied to client feedback, and one which follows steps allowing inexpensive changes in the design, to steps where changes become progressively more expensive. The idea is to encourage change in the early stages, where it can be managed with the least expense possible.

It is also important to note that FLiP is designed for the technical aspects of the project, starting at a point when you are ready to begin building an application. For many simple web projects, FLiP may be sufficient from the start of the project. For more complex projects, other research techniques may be necessary to understand the business model before starting on the application development with FLiP.

The steps in FLiP are:

1. Personas and Goals

Who will use this software? Why will they use it? A persona is a precise description of the application's user. Instead of designing feature based software, we design goal based software. This process helps deliver software desired by the end user.

2. Wire frame

Wire framing is a way to quickly model the proposed actions that will be performed by the application. The result of wire framing is a clickable model of the application that doesn't look anything like the finished project, but gives the architect an idea of how the client will help the persona achieve their goals.

3. Prototype / Front-End Development

A FLiP prototype is a clickable model of the finished application with no backend behind it. Prototyping is the largest step in the FLiP cycle, generally taking up to 70% of the project's effort. The result of prototyping is something that looks exactly like the finished application, with functionality limited to the client side. The objective is to discover exactly what the client expects from the application, and

how they want it to look. Often a tool such as DevNotes[1] is used to facilitate communication with the client about the prototype.

Prototyping is typically done in plain HTML; though if an alternative front-end technology (such as Flash) is being used the prototype may be done using that technology. The finished prototype will become the application's user interface, so the traditional sense of a prototype being a minor, throw-away version is not the case. All the effort made in this step is used in the final application.

4. Application Architecting

Once the prototype is finished, the application architect constructs the application design or schema, identifying fuseactions and organizing them into circuits. Each fuseaction's behavior is broken down into a set of fuses, and the architect writes a Fusedoc and a test harness for each fuse to be produced. Once the design has been constructed, coding can begin. There are some fantastic tools and techniques to use during this step.

5. Fuse Coding

Each coder employed on the project is sent one or more fuses and their corresponding test harnesses. The coder writes each fuse according to its Fusedoc. By using the Fusebox framework, the coder's work does not rely on the rest of the application. Each fuse can be coded and tested on its own, and can be plugged into the rest of the application. This step can be accomplished by one coder or many coders.

6. Unit Testing

As each fuse is coded, it is unit tested against its test harness. This gives the coder a way to ensure the correct behavior is produced by the fuse, without being in constant contact with the architect. It also allows the architect to employ any number of coders, none of whom need to know anything about the project as a whole in order to successfully contribute to it. However, if the development team

[1] compare @DEV

is tightly-knit or consists of only one person, unit testing creates an elegant quality control system for the basic building blocks of the application, ensuring reliability of the entire project.

7. Application Integration

As the fuses are completed, they are returned to the architect, who integrates them into the final application. Daily builds are employed, gradually transforming the prototype into a working application.

8. Deployment

Deployment is the least exciting phase of a FLiP project. Thanks to the high degree of communication between the client and architect, and the architect and coders, deployment day should hold no surprises for anyone."[1]

4.2.5.3 Ongoing prospects of ColdFusion

Macromedia[2] completed its acquisition of Allaire[3], and along with it, ColdFusion in fall 2001. Some saw the move as a strategic merger of two great companies; Macromedia with its strong emphasis on design and the user experience (Flash, Generator, Dreamweaver) and Allaire with its application-server offerings (ColdFusion and JRun). Others weren't so thrilled and felt this spelled the end of ColdFusion.[4]

A year later, ColdFusion is stronger than before and Macromedia brings out one Version after the other. Right now, ColdFusionMX is the latest product in web application developing. There are many rumors about stats but neither Microsoft with its ASP nor Macromedia with its ColdFusion ever said anything official about the spreading of their products. It looks like in the United States there are

1 taken from http://www.fusebox.org, @FUSE
2 @MACR
3 @ALLA
4 compare Rob Brooks-Bilson; http://web.oreilly.com

both technologies head on head but in Europe there are only a few CF users and practical no market at all.

The future prospects of CF are good because Macromedia, the leader in animated web applications with its Flash is working steadily on integrate all those technologies within CF and building interfaces in both products. So there might be a lot of developers which are using Flash already change their mind and switch to CF in the future.

4.2.5.4 Performance

“Most people who have heard about the faster performance of ColdFusion MX are probably wondering how in a million years something written in Java could perform better than something written in C++. Sound ridiculous? The answer is that although ColdFusion 5 was a C++ engine, it turned your Cold Fusion Meta Language (CFML) into PCODE[1], which was interpreted by the ColdFusion 5 engine during page requests. In ColdFusion MX, your CFML gets truly compiled down to Java byte code. This type of code is faster for ColdFusion MX to execute, even though the Java Runtime Engine is itself inherently interpretive. In this regard, the Java stuff is good.

The big downfall you'll immediately notice is the length of time you have to wait for your pages to compile. Even though you are writing regular CFML pages, when you save a page and request it in the browser for the first time, you won't see it until the server first finishes translating the file to Java, and then finishes compiling it. Subsequent requests to the file are super fast, but the first one is a killer. They found that the translation and compilation processes actually reduce developer productivity since a developer goes through the process of editing a .cfm file and reloading it in the browser so often. Although I haven't seen it myself, I've heard reports of pages taking 3-5 seconds to compile on fast computers while the same process in CF5 was seemingly instantaneous. Listening to your

[1] PCODE; preparsed pseudo code

hard drive churn during page development seems to be the big tradeoff for a more versatile foundation of ColdFusion."[1]

For now, the improvements in ColdFusion MX compared to the older versions (4.5 and 5.0) aren't really that important to HiP to justify the high costs for the new Versions. The JRun implementation isn't used at HiP at all and there are no plans for the near future. For those reasons and the fact that ColdFusion MX is not faster than previous versions, HiP is sticking with ColdFusion 4.5 and ColdFusion 5 for now. Even the upgrade from 4.5 to 5 makes some problems and needs a lot of improvements and changes in the code and also on the database side. The plans for the next couple months are upgrading every Server to ColdFusion 5 because it offers some new and nice features and a better server performance as well as some bug fixes.

4.2.5.5 Stability

ColdFusion has proven its high stability in the last couple years at HiP. They are using it as the only web language in production for dynamic web pages. The most time the servers run really smooth and don't produce a lot of errors. As it was used on Windows operating system it crashed from time to time. Now after all Servers are running the Linux operating system the problems with instability are almost not countable.

4.2.6 Used techniques

For the development of the CMS there were used a couple of the above technologies like described and some are modified to fit better in the HiP environment and the project requirements. First of all, the FuseBox is a lot more different than its suggested version. HiP created an own standard which spilt a project not only into fuses, every different type of file is also stored in its own folder and will help not loosing the overview.

[1] taken from http://www.webmonkey.com

4.2.6.1 Use cases and flow charts

The first step after the requirement analysis was done and the decision was clear that HiP must develop their own CMS. There were considerations how the final system could work and look like. To transfer the ideas of the development team to the participants of the meetings as well as understanding the whole software, some visual models seemed to be perfect to show how the workflow and the system could look like. Rational Rose[1] was used as a modeling tool for creating the use cases.

First there was a need of understanding what kinds of actors are participating or are part of the system:

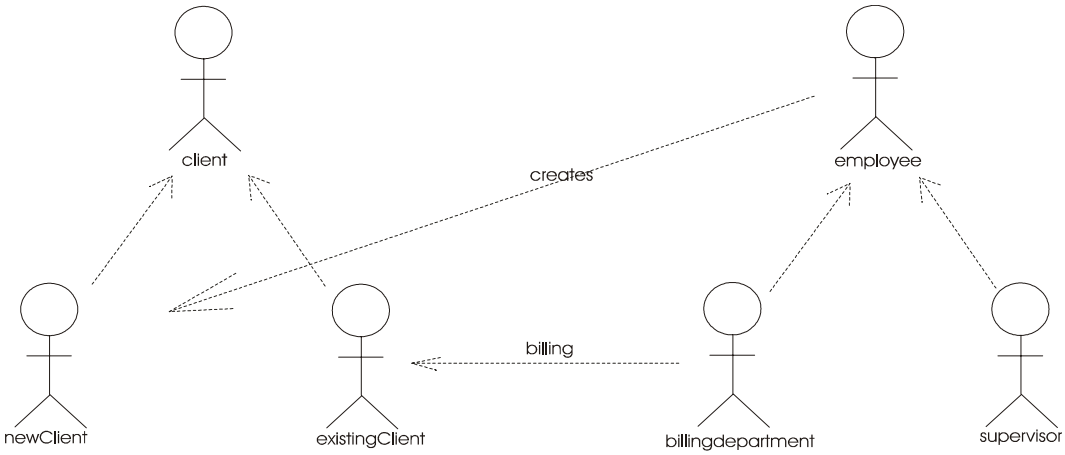


Fig. 5: Use Case Diagram, Actors

All actors as listed in Fig.5 interfere with each other. There are the employees on the side of HiP and the clients. Special employees are billing department and supervisor. Both has extended rights and features than regular employees. On the client side there are existing- and new clients.

[1] @ROSE

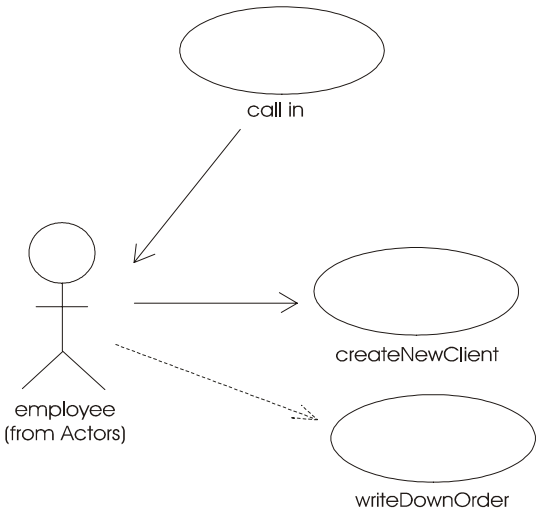


Fig. 6: Use Case Diagram, create new client

If a client calls in and wanted to have something done, the first step is to create a new client which will be stored in the database and available for further references. After the client was added, the new order can be placed.

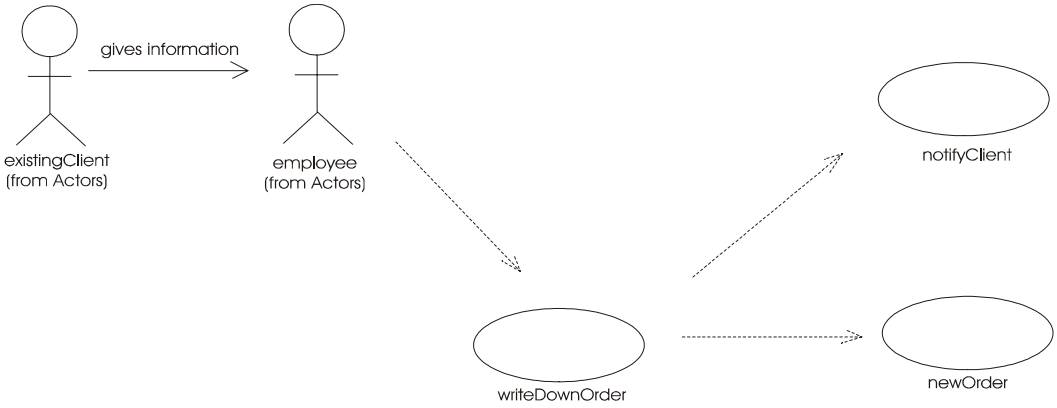


Fig. 7: Use Case Diagram, take order

Now, the existing client gives information to an employee via phone, email or in person and the employee is writing it into the system. The called action will notify the client via email and places the new order into the system.

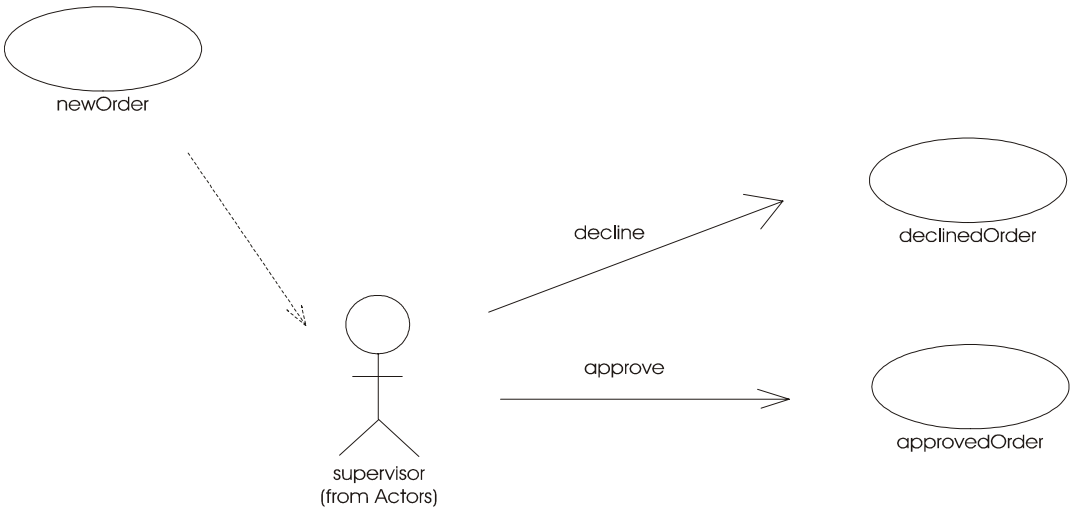


Fig. 8: Use Case Diagram, approve order

The new order is going directly to the supervisor. This actor now needs to approve or decline the order. If it is declined it will be on hold and extra action will be necessary. Usually he or she will be approving it and the new status of the order will be ‘approved order’.

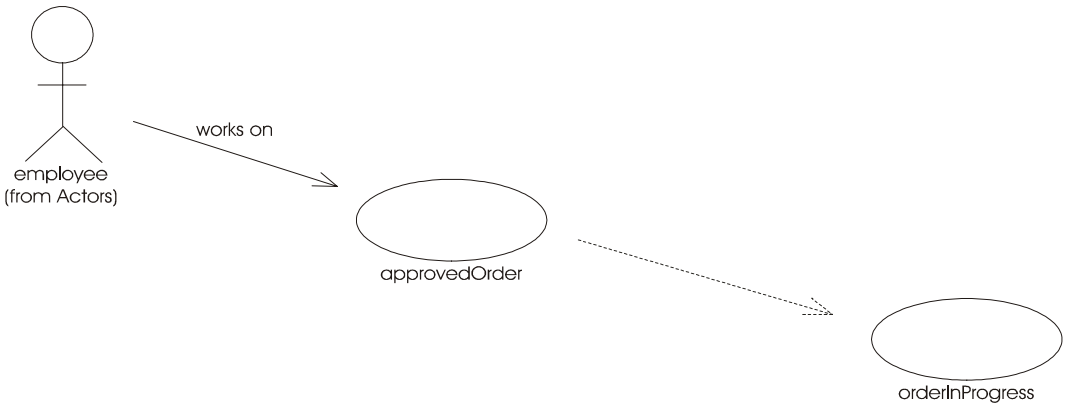


Fig. 9: Use Case Diagram, process order

The employee is working on the approved order till it is finished. Up to that point it has the state of ‘order in progress’.

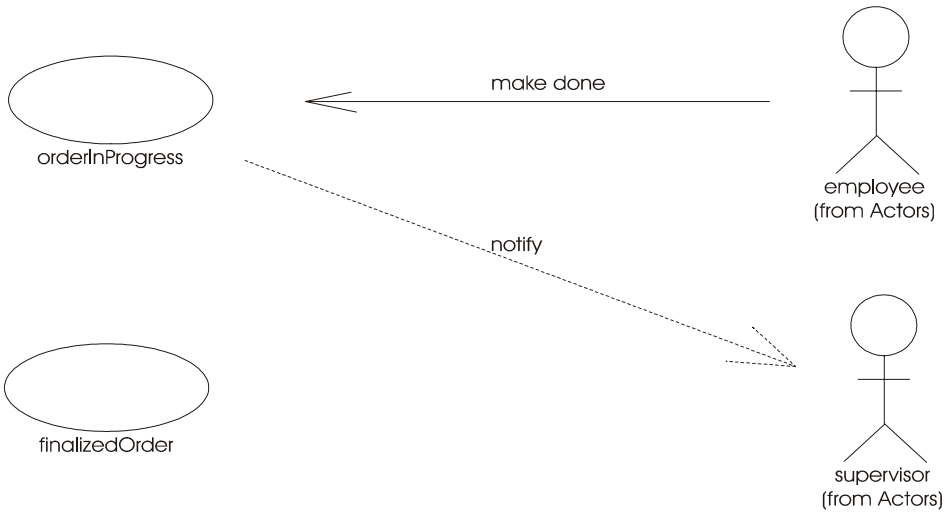


Fig. 10: Use Case Diagram, finalize order

The ongoing work progress will lead at one point that the order is done. This will change the status to done and also notifies the supervisor to approve the ‘done order’.

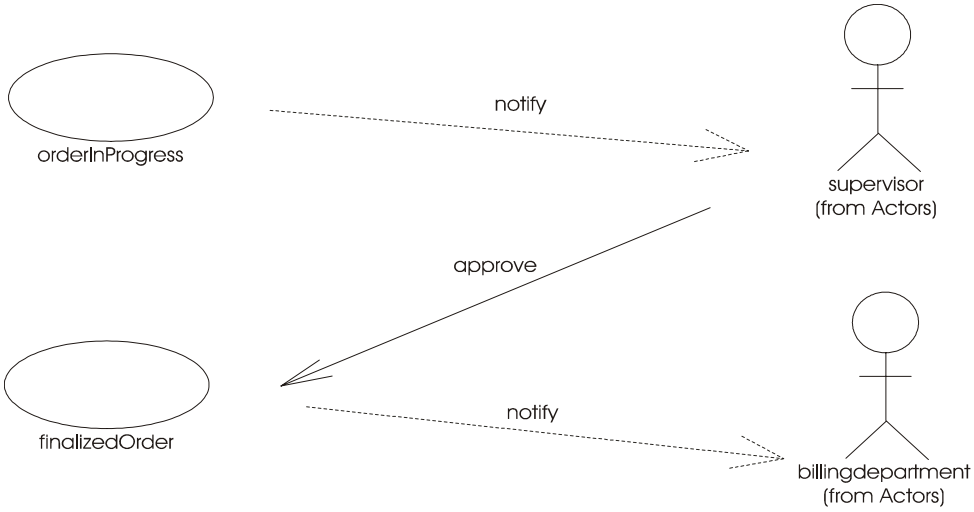


Fig. 11: Use Case Diagram, approve finalized order

Usually after the supervisor is notified he or she will check the work and then approve the order and make it final. This will lead to a notification to the billing department that there is an order to bill.

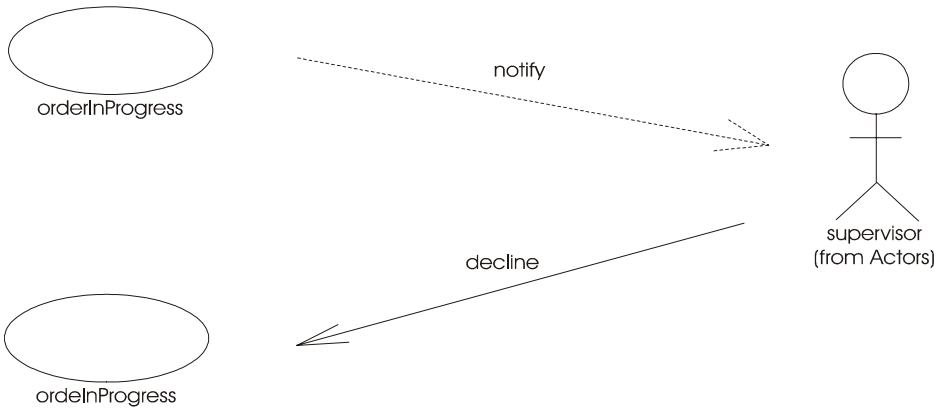


Fig. 12: Use Case Diagram, decline finalized order

However if the supervisor finds something that doesn't look right or he or she is not completely satisfied the supervisor will set the status back to 'order in progress' and it will go back to the employee.

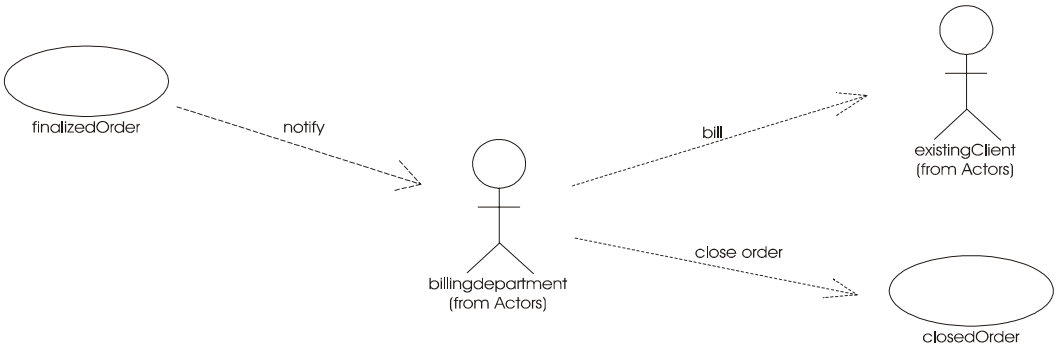


Fig. 13: Use Case Diagram, close order

If the supervisor has set the status to 'final' it will notify the billing department. The billing department will bill the client and close the order and take it out of the system.

Those documents were approved by the project team and the management of HiP and helped to understand the workflow of the system. On this basis of the use cases the wire frames were built.

4.2.6.2 Wire framing and prototyping

Wire framing was used as a way to quickly model the actions that will be performed by the application. As a result the project team got a clickable model of the application that doesn't look anything like the finished project, but gives the

architect and the management an idea of how the future system could feel. It is a step further than the use cases and helps to understand the workflow even more.

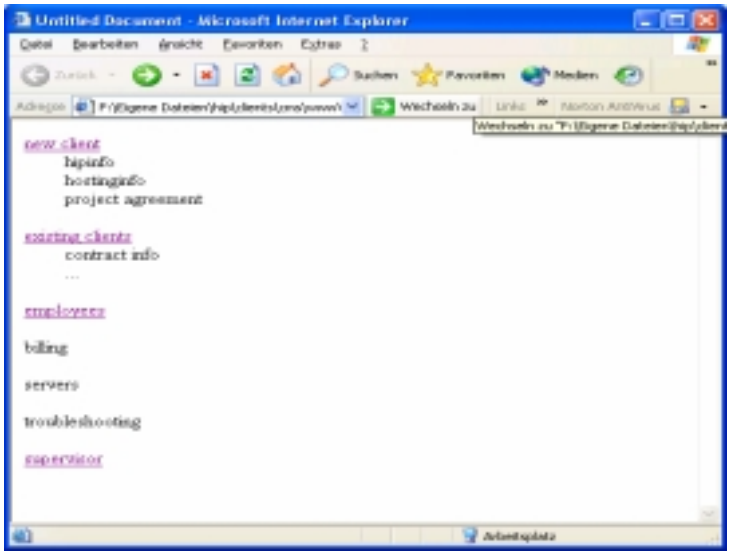


Fig. 14: Wire frame example; index page

This is a basic wire frame which shows a simple index page. It has the ability to click through the system and give the users a first impression how the workflow could be.

All created wire frames are shown in appendixes 1-11 on pages A1 to A6.

4.2.6.3 Application architecting

After the prototype was finished, the application architect constructed the application design, identifying fuseactions and organizes them into circuits. The database layout was done and created on the basis of the use cases and the wire frames. The design and layout of the database and the fuseactions has been constructed, coding started.

4.2.6.4 Fuse coding

As the project is split into small fuses, every fuse could be done by another coder or developer. They don’t interfere with each other and there is no need for a specific order of coding.

4.2.6.5 Unit testing

After the fuses have been coded, any of those were tested and checked if the needed functionality was given. This means a high quality control because each module is tested individually and no error will be undiscovered.

4.2.6.6 Application integration

After the single fuses were finished, one after another was built into the working system. It was easy to follow the understanding and test the modules after they were placed together.

4.2.6.7 Deployment

On deployment day there was only the need of copying the source code from the development area to the live server. Change some variables to match the database environment and putting it live.

4.2.6.8 Fusebox at HiP

As said before, HiP is using a special form of Fusebox. This form is slightly modified from the original suggestion of Fusebox.org and it fits better in the HiP environment. The easiest form of CF could be something like the following:
The programmer develops his pages as he did before, and whenever he needs some dynamic content he embeds a ColdFusion-Tag. The following Example prints out the date on a simple page:

```
<CFOUTPUT>
#DayofWeekAsString(DayOfWeek(today))#, #MonthAsString(Month(today))#
#Day(today)#, #Year(today)#
</CFOUTPUT>
```

Fig. 15: Listing, easy ColdFusion example

But this has nothing to do with real programming and the use of FuseBox. As said before it derivates the functions called fuses into separate closed snippets. The description of the used technique will be done on one example of the final CMS. There are some modules. Each module is in a separate folder. Every folder has its own file from which the fuseactions are called. The modules for example are clients and supervisor. They don’t interfere in any way and could be insulated from

each other. So there is a folder for every module:

```
<ServerPath>/cms/clients
<ServerPath>/cms/supervisor
```

Fig. 16: Listing, module description, server paths

In the subfolders the different files are derivated between their purposes. There are query-files for querying data from a database only. They are stored in the folder ‘qry’ and are called ‘qry_sample.cfm’. The display pages for outputting the dynamic content are stored in the subfolder ‘dsp’ and are called ‘dsp_sample.cfm’. There are also action files which are small applications and write something to the database or execute some actions. They are found in the folder ‘act’ and are called ‘act_sample.cfm’.
The control file is called ‘index.cfm’. In that file, all the fuseactions are described and call the templates for executing the functions.

```
<CFSWITCH EXPRESSION="#fa#">
          <CFCASE VALUE="home">
                    <CFINCLUDE template="qry/qry_all.cfm">
                    <CFINCLUDE template = "dsp_home.cfm">
          </CFCASE>
          <CFCASE VALUE="search">
                    <CFINCLUDE template="qry/qry_search.cfm">
                    <CFINCLUDE template = "dsp_home.cfm">
          </CFCASE>
                    <CFCASE VALUE="addclient">
                    <CFINCLUDE template="act/act_addclient.cfm">
                    <CFINCLUDE template = "dsp_addclientcontact.cfm">
          </CFCASE>
</CFSWITCH>
```

Fig. 17: Listing, control file

As you can see in the above example, in every case, called ‘home’, ‘search’ and so on - are the files included and executed. This is the basic technique of FuseBox. To add more flexibility to that model, HiP changed it for this project and used instead of display-pages, templates to build the appearance of a site. This has the advantage of switch only that file, and the site will look completely different. There are also headers and footers. So all the basic HTML tags are only in one file and are included as needed. This means fewer files, less storage and easier main-

tenance if something changes. Global or sub global variables are also stored in files called ‘app_global.cfm’ and ‘app_local.cfm’. They are also included in every control file. A complete set of files for one application would look like the following:

```
---------- index.cfm -----------
<CFINCLUDE template="../app_global.cfm">
<CFINCLUDE template="app_local.cfm">
<CFSWITCH EXPRESSION="#fa#">
          <CFCASE VALUE="home">
                    <CFINCLUDE template="qry/qry_all.cfm">
                    <CFSET mainbody = "dsp_home.cfm">
          </CFCASE>
          <CFCASE VALUE="search">
                    <CFINCLUDE template="qry/qry_search.cfm">
                    <CFSET mainbody = "dsp_home.cfm">
          </CFCASE>
          …
</CFSWITCH>
<CFINCLUDE template="dsp/dsp_template.cfm">
```

Fig. 18: Listing, complex control file

```
----------- dsp_template.cfm ------------
<CFINCLUDE template="dsp_header.cfm">
<CFINCLUDE template="dsp_sub_nav.cfm">
<CFINCLUDE template="#mainbody#">
<CFINCLUDE template="dsp_footer.cfm">
```

Fig. 19: Listing, easy template file

This is just a simple template file. The appearance of the site would look like this:

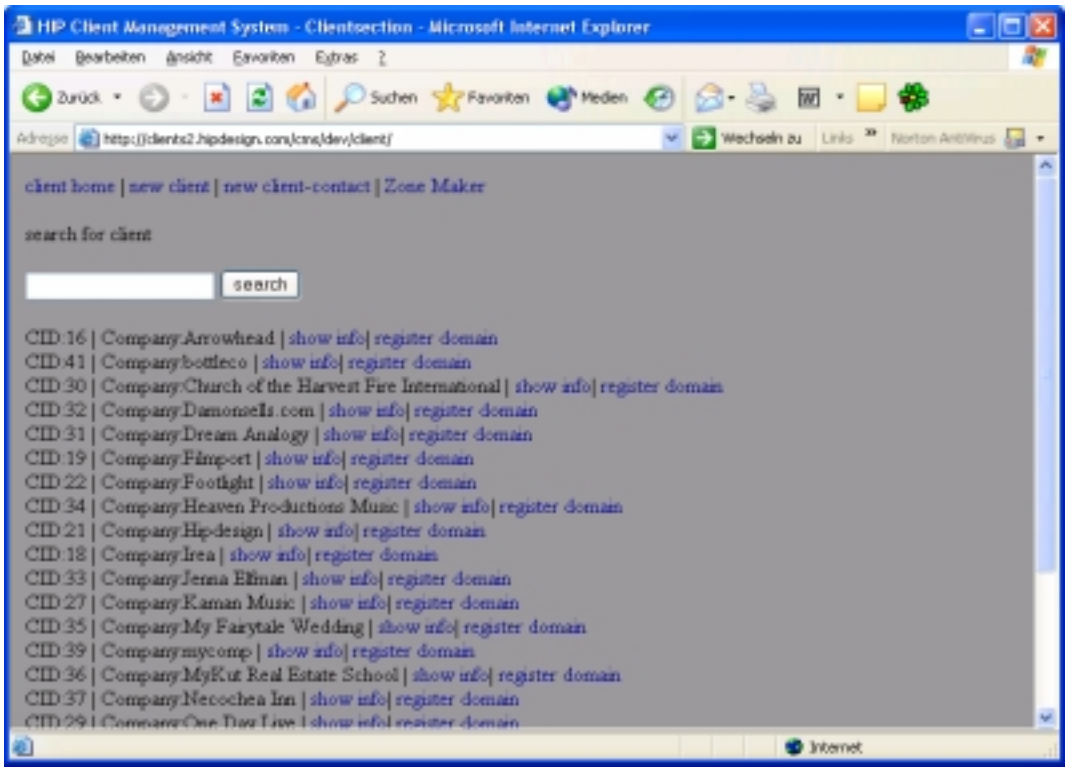


Fig. 20: Screen of the example template file (of Fig. 19)

So it includes a header, a navigation bar, the display page called #mainbody# and the footer. It is easy to see, when you change this file, add tables, use colors or different fonts and styles you could change the whole appearance of the system within minutes.

To demonstrate this behavior there are two examples which show how easy it is to alter the template file and change the appearance of the displayed page.

```
-------dsp_template1.cfm--------
<CFINCLUDE template="dsp_header.cfm">
<table width="100%" border="0>
  <tr>
    <td><CFINCLUDE template="dsp_message.cfm"></td>
  </tr>
  <tr>
    <td><CFINCLUDE template="../../dsp_global_nav.cfm"></td>
  </tr>
  <tr>
    <td><CFINCLUDE template="dsp_sub_nav.cfm"></td>
  </tr>
  <tr>
    <td><CFINCLUDE template="#mainbody#"></td>
  </tr>
</table>
<CFINCLUDE template="dsp_footer.cfm">
```

Fig. 21: Listing, template1 example

The first template file (dsp_template1.cfm; compare Fig. 21) produces the following display page:

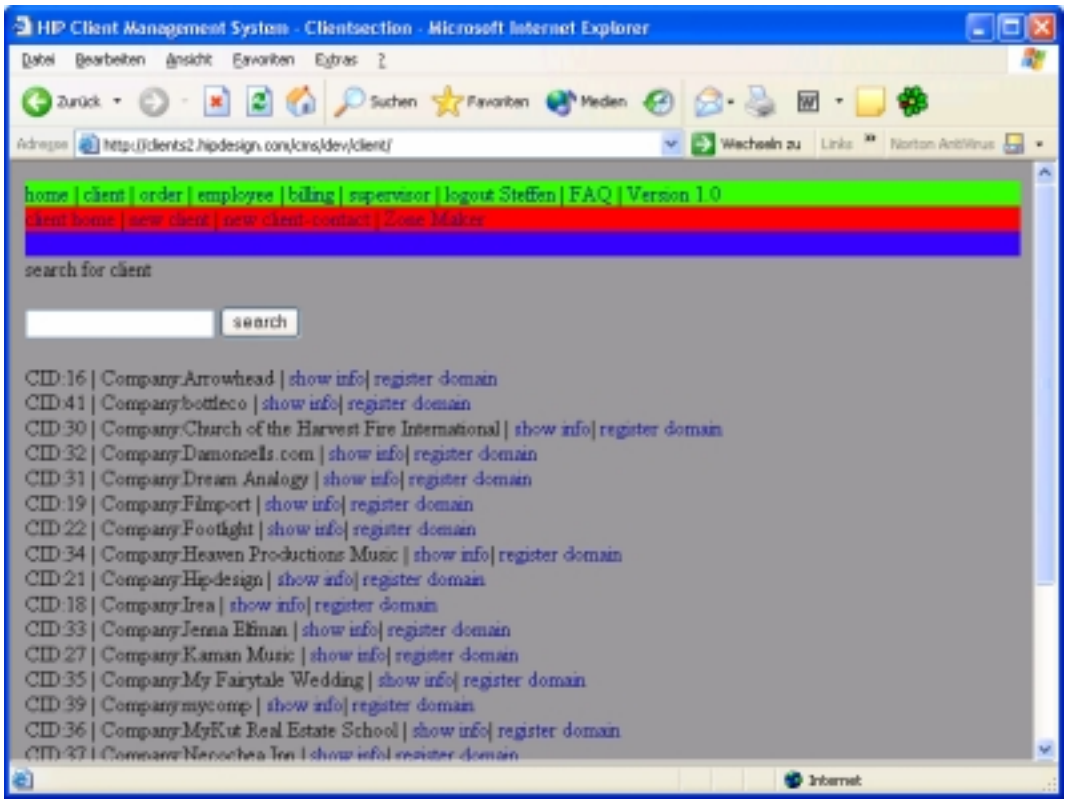


Fig. 22: Example template with navigation bar on top

```
-------dsp_template2.cfm--------
<CFINCLUDE template="dsp_header.cfm">
<table width="100%" border="0">
  <tr>
    <td bgcolor="#FF0000"><CFINCLUDE template="dsp_message.cfm">
</td>
  </tr>
  <tr>
    <td><table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td width="8%" valign="top" bgcolor="#33FF00">
          <CFINCLUDE template="../../dsp_global_nav.cfm"></td>
    <td width="8%" valign="top" bgcolor="#3300FF">
          <CFINCLUDE template="dsp_sub_nav.cfm"></td>
    <td width="84%" valign="top">
          <CFINCLUDE template="#mainbody#"></td>
  </tr>
</table></td>
  </tr>
</table>
<CFINCLUDE template="dsp_footer.cfm">
```

Fig. 23: Listing, template2 example

The second template file (dsp_template2.cfm; compare Fig. 23) produces the following display page:

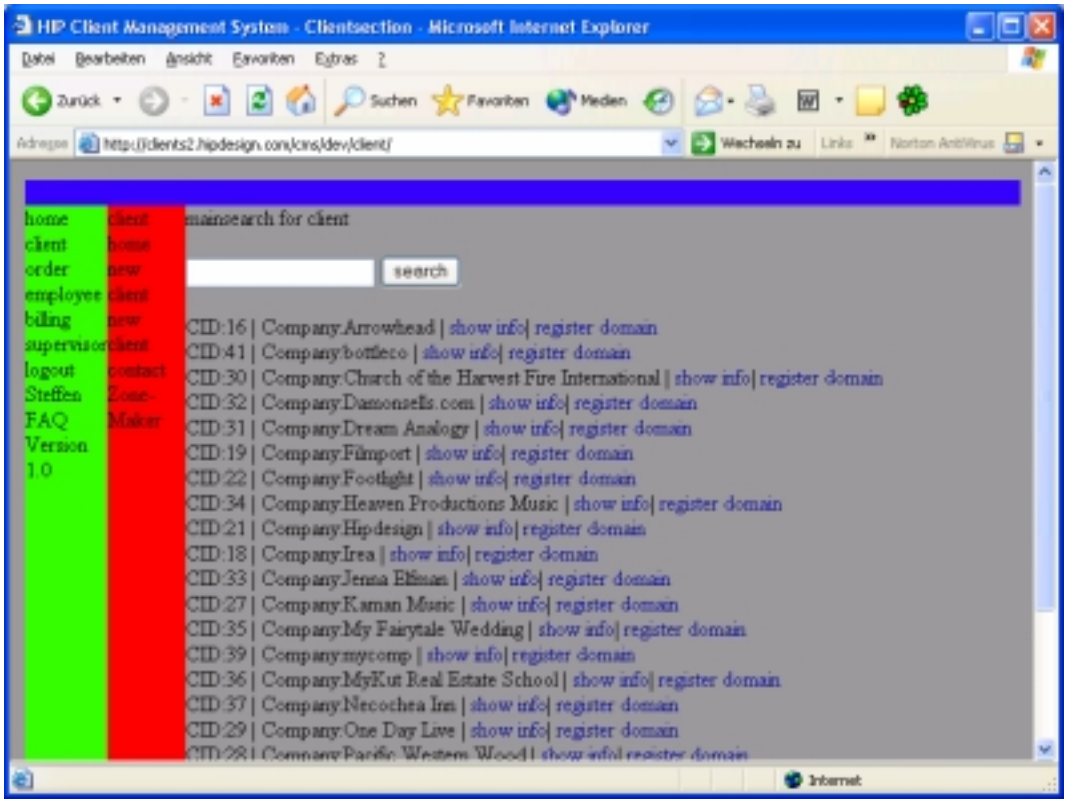


Fig. 24: Example template with navigation bar on the side

4.3 Comparison product requirement specifications with final CMS

After the project was finished and ready to going live first there was a test for the required features as listed in the product requirements.

4.3.1 Test for planned features

The best way to test if everything is implemented and working properly is to take some cases out of the regular daily work, put them into the system and check if there are any difficulties or additional things to do out of the view of the project team.

Also, there should be more complex tasks in every task field. As a basis for testing there are the earlier mentioned flow charts and use cases which will be followed. If there are no problems, all required features are implemented correctly.

4.3.1.1 Generating sample tasks for test purpose or take real ones out of ongoing projects

A1) An existing customer calls and wanted another email account set up.

A2) A customer has problems login to his ftp (forgot the password)

B1) A new customer wants his domain registered, email, ftp and server set up and move his existing web site to HiP servers

B2) A new client wants the whole package. Design, development, hosting and afterwards maintenance with regular site updates. There are a couple people on both sides involved. One person from the client is responsible for the technical part another for the artistic part.

C1) An existing customer wants a new design/layout for his website and some new dynamic features

C2) An existing customer sends some files with tables and images to upload to their website.

D1) A client wants to know the progress of his order and lookup his status.

E1) An employee wants to know how many hours he or she spent in the last month

F1) A new employee joins the company and needs an account for the CMS

G1) A client calls with some basic problems concerning email and/or ftp

4.3.1.2 Putting the sample tasks into the system and report any issues as well as the workflow

A1) An existing customer calls and wanted another email account set up:

As it is an existing customer, there is no need to put in additional information to the database on the client contact. The test starts straight at *order/new order*. There needs to be the company chosen as well as the client contact.

The next step will be picking the editor from the employee list who will be responsible for that task. In this case, there is no need for a rush and no need for a supervisor to approve the order or the finalization of it. The employee task list the new order appears. The editor (employee) goes into it, looks at the description and does the work as usually and listed. After the job is done, the employee puts in some additional remarks if applicable, the spent hours and the type of the task (here: server). The last step on the editor's side is to finalize the order. This final order will appear in the *billing section* and the billing department has the ability to review the task, read any notes or check back with the editor. If everything is clear they will take the task out of the system and put it into the separate running billing system. The order is now billed.

Remarks: no problems on this case

A2) A customer has problems login to his ftp (forgot the password):

This order is very similar to the previous one (A1) with some changes: As they can't login, it is obvious that it is urgent and needs to be done immediately. Since there is no password stored in the database and there is no live access to the passwords on the server, there is a need of changing the password. After this is done, the new password is stored in the CMS database in encrypted form and is passed to the client. If the client calls the next time with the same problem, the password could be looked up at the system. The rest of the task is the same as A1. For testing this case was putted through the system but won't be discussed here.

Remarks: Password could not be looked up, after changing it on the server and stored it in the database, the case was normal without any further problems. There is no function to change user passwords as listed in the requirements.

B1) A new customer wants his domain registered, email, ftp and server set up and move his existing web site to HiP servers:

As it is a new customer, there is a need of storing the contact information of the company as well as the client contact in the database. For this purpose there is *client/new client* where the company information is put in as well as the client contact personal information. Then there will be a new order added into the system. As it is all server related things, there is only one person involved in this task. So it will be as simple as the ones before. The only change in the workflow is that a new customer usually requires a supervisor of HiP to approve the client and the order. So while putting all the required information to the *new order* the supervisor needed button needs to be checked. After processing it won't go to the editor in the first place. It will go to the supervisor with special remarks and he or she needs to approve or decline the order. If the supervisor approves the order the task is going through as usual and listed in A1. If the supervisor declines the order, he or she needs to contact the client and talk about it or just delete the order.

Remarks: no problems in this case.

B2) A new client wants the whole package. Design, development, hosting and afterwards maintenance with regular site updates:

There are people on both sides involved. One person from the client side is responsible for the technical part another for the artistic part.

The first step is as on B1. But there will be more people on the client side involve so there is a need of adding additional client contacts to the system. After the company and one client were added, the employee needed to go to *client/add client contact*, choose the company from the list and add the additional persons to the company. As this whole order is more complex than every previous order there is a need of some planning in advance, but this is not a task of the CMS in this case and will be taken as already done. However, the planning could be

placed in the CMS like every other task. There are a couple tasks in this project which will be placed on different editors schedules. It is the same process for every task. For example, the technical part will be associated to a technical editor, the artistic part to somebody else. The next steps are similar to previous case studies. While everything is going through the system one after the other task will be finished. After they are done, they will appear in the *billing section*. The billing section needs to check back and then bill the client.

Remarks: The billing department has the biggest problems here. It is not obvious for them to see after one task of the project is final, that they can not bill the client for the whole work. They need to wait for every part to come in and to be final. There is a lot of coordination necessary.

C1) An existing customer wants a new design/layout for his website and some new dynamic features:

As the client is already in the system, it will go straight to *order/new order*. On this project are also two different departments at HiP involved - the technical and the artistic. So the project needs to be split into some parts for the individual editors. As this is also a larger project there is a need for a supervisor; it will be processed through the system as usual but will be discussed here in detail. The billing department has also the same problems like on B2 because there is no feature to attach different orders to each other.

Remarks: The billing department has problems again; no clear action to bill after one part of the project is done.

C2) An existing customer sends some files with tables and images to upload to their website during their regular maintenance (CSC):

Like before, there will be a new order placed in the *order/new order section*. The CSC- checkbox will be activated and the only other thing is that the client sent some files which need to be uploaded and are part of the project. The CMS offers a feature to upload files and attach them to the order. The only limitation is that there is only one file allowed at the initial order placement. Afterwards there can

be as many files uploaded as necessary. It doesn't make to many difficulties. The only thing is that after the order is placed the employee needs to go into it and upload the remaining files to the system. Everything else will be handled as before. It doesn't make any difference for the editor if the order is on a regular invoice/billing or if it is a free update of their maintenance contract. The differences here are on the billing department side. The CSC orders are marked as on of those and the billing department only needs to see that there was activity for that client. They will take out the order from the CMS and make a note on their separate system that one of the free updates is used.

Remarks: Multiple files in the initial order placement are complicated.

D1) A client wants to know the progress of his order and lookup his status:

There is no feature implemented where the client could check the order status on their own. They need to call in to ask what stand the project is on.

Remarks: No feature at all!

E1) An employee wants to know how many hours he or she spent in the last month:

On the employee page, the employee could go into his *personal site*, check and filter his or her spent hours during a period of time. It will automatically display the hours of the current month. The billing department has got the same feature and could check the hours of ever employee.

Remarks: implemented as listed in the product requirements.

F1) A new employee joins the company and needs an account for the CMS:

The superuser which is a different type of user who has every right in the system needs to add a user to the system. Therefore he or she goes to the *system settings* in the *supervisor section* and adds a user. The superuser needs to assign a username, password and the role (billing, employee, supervisor, superuser, guest or a combination of them) for the new employee.

Remarks: No problems occurred; the feature seems to be working properly.

G1) A client calls with some basic problems concerning email and/or ftp:

For that case, there is a need to place a new order only for statistically reasons. There won't be any billing for that. So after adding hours for the task, the editor needs to change the price for that to 0. This is possible in the review order and task sheet. As it is a simple problem which probably occurs on a regular basis to other customers as well, it makes sense to put it into the *FAQ section* and make it available for other clients and refer to it for future reference. In the *FAQ section* there are categories in which the problem could be placed. If nothing matches, a new category can be created; it should be placed on the live side and available to every client.

Remarks: No problems on the backend while adding the problem to the database, but there is no live site to date.

Summary:

Any of the above cases could be processed through the system. However there are some open points and difficulties, let's look at them again:

- There is no password lookup on the server available
- The billing department can't bill problem free if the project is split up into different parts, it need a lot of work and accuracy to keep everything attached to each other.
- Multiple files handling only available after initial order placement
- No section for the client where they can check out their information
- No front end for FAQ, only storage in database

4.3.2 Deviations and Aspects

There were only minor problems during the test period. Almost every function worked as wanted. However, to make it a complete system, there is some work to do in future versions. Some of the points might be implemented some others aren't.

The password lookup was taken out of the system. HiP built their servers in a certain way and does not use a database on the server side for password management. The passwords are stored in the standard flat files. To access or change those files, the user needs root rights. Linux doesn't offer a feature to lookup passwords at all. So, if somebody wants to change passwords during a web interface, the web server and/or ColdFusion server need to have root rights[1], too. This would be a too large security hole. If they will switch in the future and store their passwords instead of in the flat files in a database, the feature could be implemented and used as wanted. For now, there is a fix which isn't optimal at all but will help. If a password is changed on the server, it needs to be changed on the CMS database manually, too. So, if it is entered in there, it could be reviewed and in the case of loss given to the client.

If there are larger projects and more people involved in one and the same, it is difficult for the billing department and/or the supervisor to keep everything in mind because there is no logical attachment between those 'individual orders'. There should be a way to split or join projects and if only one part of it is done, it doesn't appear to be final. It would be also much more comfortable to update and delete, or bill them if only one global order needs to be taken care of. As a matter of fact, HiP is a small company and almost every employee knows what is going on. Especially the billing department knows if there is a large project going on and if they see the client name they can probably see that there are more things to do and wait till everything is done. The only problem here is that the billing department isn't specialized in the project work and different fields of the company so

[1] root is the user on a Linux system who has access to every functionality

sometimes they need to talk back to the departments if there are any problems in understanding.

If somebody sends multiple files at the initial order placement, there is no possibility to add them to the system because in that situation it only allows one file. After the order was placed, there will be the functionality to add as many files as needed. In the most cases there won't be any file at all to be attached to the order. It is difficult to upload multiple files before the order is placed (it is technically that way because before the ‘add order – button’ isn't clicked there won’t be a physical order) there is a workaround suggested. If there is the need for more than one file, they should be compressed with a packing program like ZIP and joined to one file. It won't be necessary to implement that function in the future. It would be nice but isn’t mandatory.

In the beginning of the project there was the target to implement a function where the client could log on and check different aspects and data of their projects. But after thinking of it, this would mean a lot more administration expenses because every client or client contact needs al log-in and they probably need some training to get comfortable with it. The gained information out of the system however wouldn’t be that helpful at all. So the only remaining feature would be the ability to update their contact information or see who is working on the project. But as a matter of fact, the contact information doesn’t change in the most cases at all over a very long period of time and the client will receive a order confirmation after placement via email where they can review the editor and every other data. This feature will be still on the project list but doesn’t have a high priority.

The FAQ section is actually built as it was on the requirements list. So there won’t be any need to improve or change that. However there is no front end where the clients could review the information given by HiP to them. Somebody at the company will be working on that task in the near future and it will be implemented to HiPs website. So it isn’t really a missing feature of the CMS. It is something more detailed and needs to be implemented outside of the system.

4.3.3 Discussion of new aspects

Sometimes functionality loses its priority. This happened to the client log-on feature. In the first meetings HiP thought this would be a really nice function and it needs to be in the final version of the CMS. But as the research has shown, some of the clients would not use it nor need it at all. They need to do many other things than their basic business. In that time they won't be productive and this would mean either that they will need more people or the existing employees need to work more.

After the first beta version was launched the employees needed to test the system with some simple tasks out of their daily work. Everybody made suggestions how some tasks in the system could be improved or made much easier. The most of this was implemented as good as possible in the final version (1.0). Everything that couldn’t be considered will be probably implemented in one of the future versions.

The workflow is in the most cases an abbreviation of the old-style how it was done before there was the CMS. After working a while with it, there might be some improvements to the modules and this could be considered also in the future.

4.4 System description of the CMS

After the system was built and the test of the required functions was done, a closer look at the features and functions as well as the expendabilities for future versions is examined.

4.4.1 Features / Functions range of the CMS

The CMS in the final version consists out of eight directly accessible modules and one system module which cannot be viewable in the browser. It is only for automated tasks.

The accessible modules are:
- Home
- Client section
- Order section
- Employee section
- Billing section
- Supervisor section
- Login/logout
- FAQ

To understand the workflow and the terminology in the following description first there should be some explanations. To use the system, there is the need to log in. The system differentiates between roles. The most powerful user in the system is the "superuser". Superusers can go into every module and have the power to add and change users and their details. Superusers are also members of supervisor and billing. Another important role is the supervisor. Supervisors are usually normal users with the ability to approve, decline or delete orders (in progress) or completed orders. They have a special module for those tasks. Then there is billing. Billing users have access to the billing section and can only review the orders and bill them. Another role in the system is employee. Every user of the above is also an employee. An employee has access to the modules for order, client and employee. In the daily work every user of the system acts as an employee. The last

but not yet used role is called guest. Guests are clients or other people who will have access to the system but won't be able to go into one of the above modules. This is reserved for further versions.

The system has built in some automatic functions. Every time a new order is placed, the editor of it as well as the client will receive an email with all necessary details. If a supervisor needs to approve the order, he or she will also receive an email. If an order was processed and is going to be done, the supervisor will receive again a status email. If the order will be finalized and ready to be billed, the billing department will receive an email. The system sends a list every week of all open tasks to each employee where they can review their schedule. Another function sends emails to the editors with open tasks reminders if the status of a job becomes very high and the completion date is near.

The also existing auto priority function will be reviewed in paragraph D.

A) Home

On the home page it will display a personalized overview over the system. It depends on the role of the user which modules and links he or she will see. The same applies to the navigation bar which is totally personalized. The version history is also available in the home section.

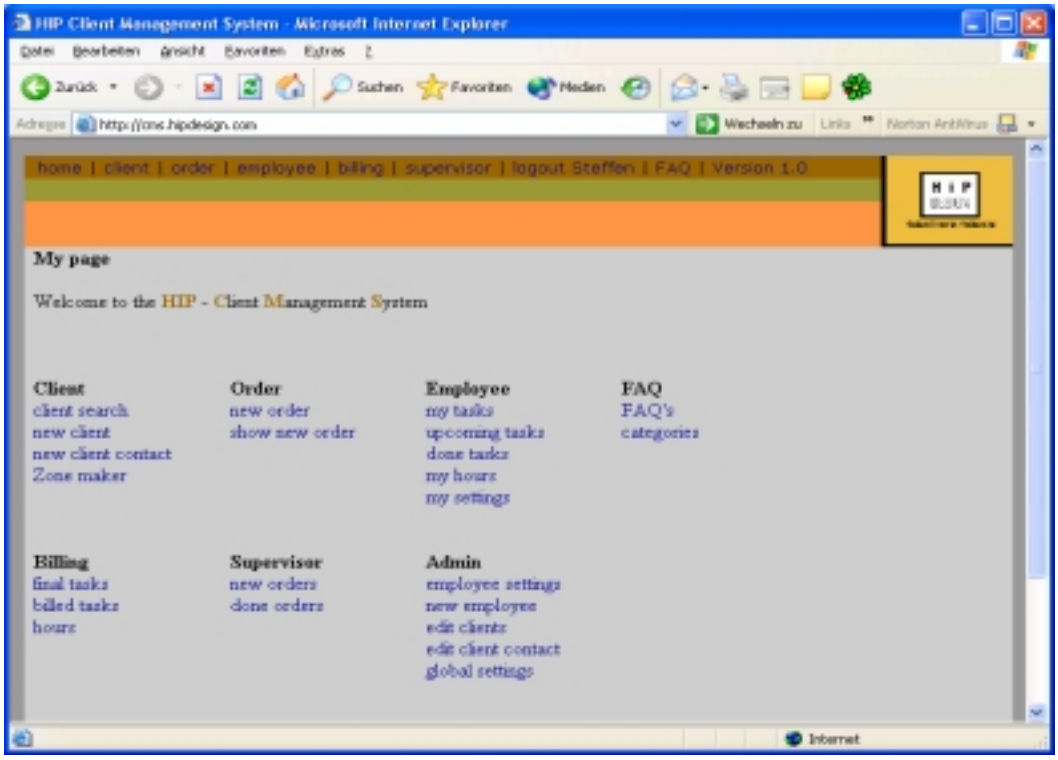


Fig. 25: Screenshot of CMS, home (for supervisor)

B) Client section

In the client section are four sub categories. On the client home you can search for clients, review the details, change their status to 'Client Service Center member', lookup what email addresses and usernames they use and of course update the contact information and client contacts.

The new client will add a new client to the database; because a client also needs personal information it will lead to adding a client contact after the company is entered. There is also a separate function for adding multiple people to a company.

A special feature is the zone-maker. If a new client needs to be set up on the server, there are a lot of entries like DNS, email settings, web server and ftp necessary. This simple script produces all those entries in the correct form.

C) Order section

The order section basically only adds new orders to the database. This is one of the key features of the whole system. There is a review for new orders and the ability for a supervisor to quickly review and approve or decline a new order.

D) Employee section

In the employee section there are all personal open tasks listed in the order of their priority. At the time of the order placement, the employee needs to set a priority for every task. If the end date of a task is near, the system will automatically set the priority new and it will move higher and higher in the task list of the employee. How the auto priority acts is set in the system settings. If there are e.g. to finish a project 10 days left, it sets the new status at least to the 4th highest priority. If there are only 7 days left, it will move it to priority #3.

The employee is going into the task details for adding additional information, work hours, upload files and change the status after the task is done. There is also a feature for reviewing done orders and check if there any upcoming tasks which aren't jet approved by a supervisor.

Behind the 'my hours button' is the functionality of reviewing all booked hours of the employee. They can filter by date.

Some more important functions are the setting of the employee details. Every employee can go in and change their username and/or password, email address, name and the ability to receive system mails.

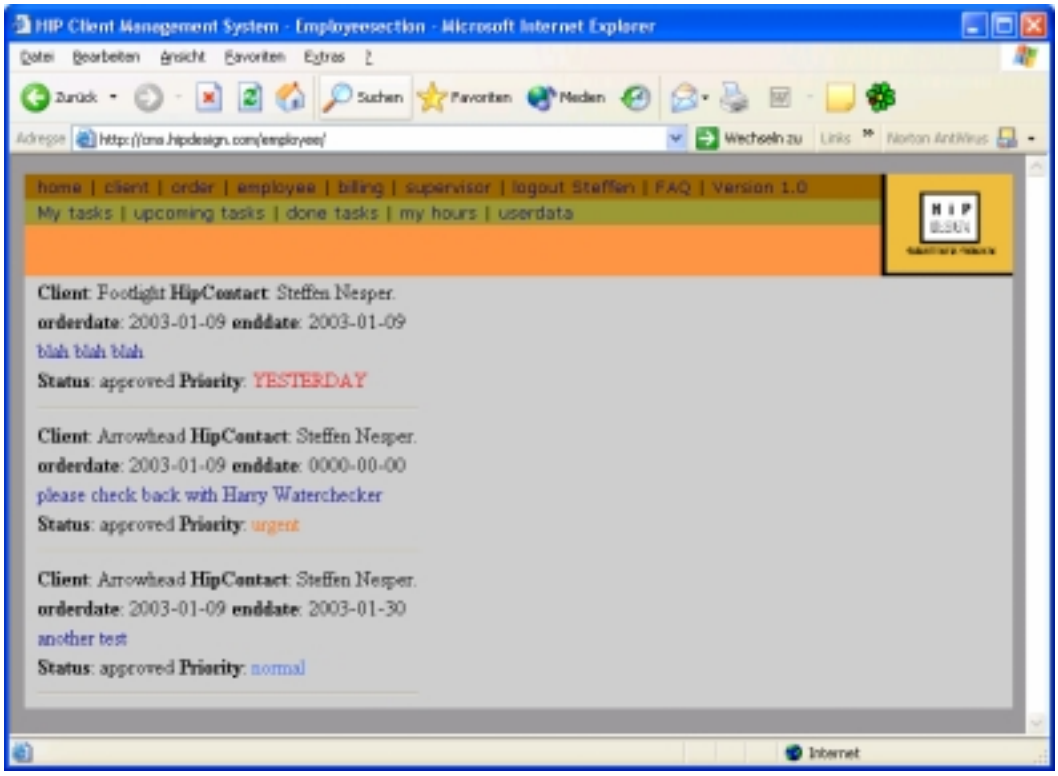


Fig. 26: Screenshot of CMS, employees task list

E) Billing section

In the billing section are only some basic functions. Members of billing can review final orders and take them out of the system. They can also review the hours of every employee who worked on some tasks in the system.

F) Supervisor section

The supervisor section is only for new orders, done orders and links to the special settings menu. If there needs to be something reviewed or approved, the supervisor will find those orders in here. Supervisors also have the ability to filter for all other orders and review and delete them. The special settings menu (compare paragraph G) is only available if the supervisor is also a superuser.

G) Supervisor special settings

In the special settings menu the superusers can create new accounts for the system, change their roles, alter all other data, review clients and client contacts as well as review the system settings. In the system settings are all information available about the implemented and automatic functions.

H) Login/logout

If somebody goes to the URL of the system, he or she will be redirected to the login page if they aren't already logged in. It is just a basic user authentication.

I) FAQ

The FAQ section is for frequently asked question. If a client calls and needs some assistance with a common problem, the employee could place the question and the solution into the FAQ section and refer the next time the problem occurs to that page.

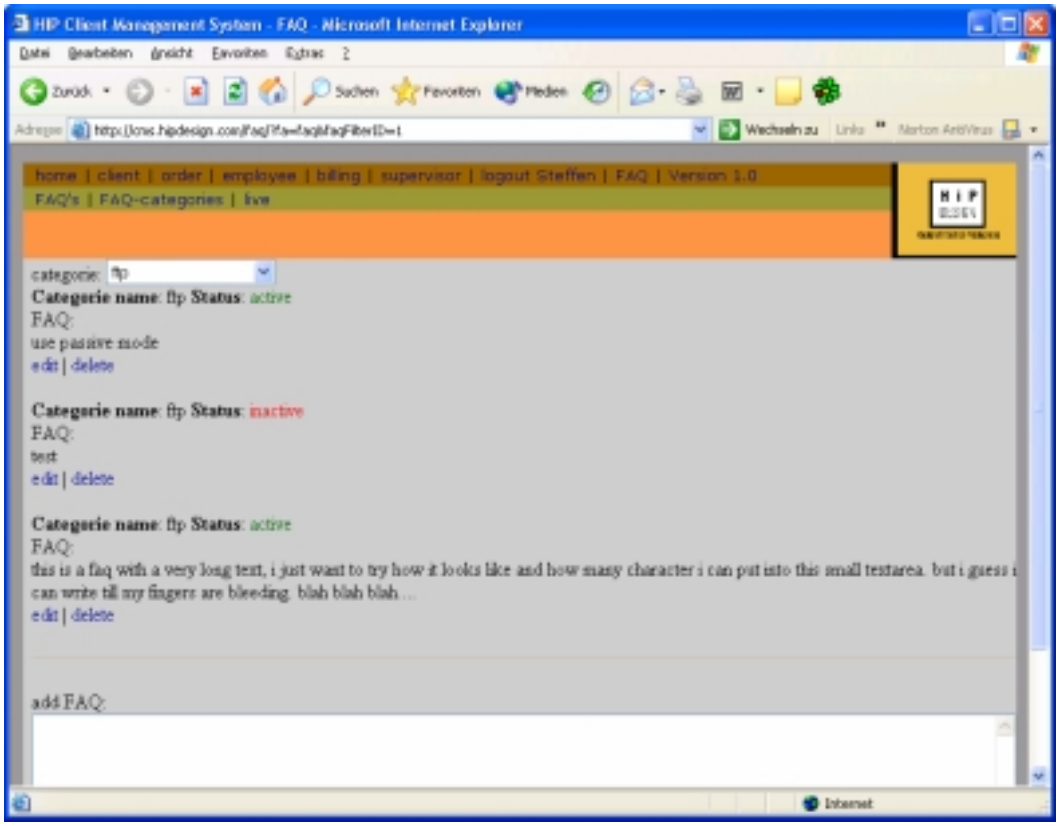


Fig. 27: Screenshot of CMS, FAQ section

4.4.2 Expendabilities and future plans of the CMS

The following points are open and could be implemented in a future version

- Create a client section where they can log in (lookup for all relevant data)
- Add a feature to split and/or combine a project
- Multiple people working on the same order
- Create a news section as a global information platform for all HiP employees and clients
- Create time tracking for modules (when was it used the last time, automatic functions)
- Read the web server control files (httpd.conf/apache.conf) out of the system (like it is done with the email files) for easy server setup and maintenance.
- Read the ftp server control files (/etc/group) for ftp tracking
- Update the system access to be more secure; disable cookies and switch to session management

The implementation of a client section is still not mandatory for further versions. If there are new aspects in the future, more ideas will be implemented in the CMS. Also, there might be the chance that a client section is coming. Right now there is no need for that and it is not practical to implement this feature because it will lead to a lot more administrative process. A much higher priority has the need of splitting projects after an order was placed or placing different task in a container and marc it as one project. So if only one part is finished, it will not appear in the billing section and it would be clear for everyone that this project consists of multiple tasks for different people. A news section could be something really need because there is no internal mailing/emailing system running. So whenever you need to inform everybody in the company, it could be a place to go. In addition to the news section, there are internal message boards that can improve the system and lead to a global information and working platform. At the moment, there are a few functions as mentioned before that are running at certain times. There is no control if it is still running or if there are any errors. This should be stored in a database and also posted somewhere in the system if something goes wrong. This might also be posted within the news section.

Another idea during the development was reading the web server configuration files out of the system. The reading process would not be difficult because it is more displaying the information and attach it to the right client. There are too many difficulties to implement that within a short time. But this could be done in a future version as well. The same thing is also applicable for reading the ftp server configuration files. It might be easy to put it in the database but to really make it usable it needs a lot of programming. The security system or user management and login is based on simple passwords and with cookies stored on the client machines. It will store an encrypted access level in the cookie but if somebody will be able to hack that key, he or she could be doing everything in the system. So there is a need of a stronger mechanism and probably changing from cookies to session management as well.

5 SUMMARY

After a final view and with a little bit of distance I think I can say that I am proud to finished the project within the given time range and lead it to a success out of my view.

Sometimes it was difficult for me to place my concepts and opinions in the project team as well as keep the schedules. The main reason for that was that I was the only one who actively worked on the project. Everybody else in the project team only had a supporting role. In between I often needed to do some other small tasks or projects. But this was also a good learning experience for me because this is how it will be in a proficient environment later in my professional life.

The analyze phase especially of the off-the-shelf products wasn't satisfyingly at all for me. I thought there might be some products which could fulfill the needs of HiP and are on the market which I could have been tested. Also the support of the tested systems sometimes wasn't able to serve all necessary information. Sometimes it seems that they didn't have interests in selling their products. As said before, the management of HiP wanted from the beginning that we are building our own system but if there would be something usable they were ready in every phase to take a closer look at it and if it could fit the needs of the company this could have been the way to go.

In the five months at HiP in the United States I have gained the ability to work in a professional environment and they supported me whenever I needed support. I would say that I also was able to use a lot of my learned techniques during this project which I gained during my studies at the University of Applied Sciences in Constance.

I am really glad that I made the decision to go abroad writing my thesis. It gave me a great foreign learning experience at the end of my studies and I got some

more knowledge how a project of this size will be handled in another cultural environment.

The next version of the system is already in planning and will include a lot of the above mentioned functions.

ABBREVIATION LISTING

ASP	Application Service Providing
CEO	Chief Executive Officer
CMD	Client Management Database
CMS	Client Management System
CRM	Customer Relationship Management
CTO	Chief Technical Officer
FAQ	Frequently asked questions
HiP	Haaland Internet Productions or Hip-Design
ISP	Internet Service Provider

CAPTIONS, FIGURES, TABLES

SOURCES

@ACSC	http://www.macromedia.com/support/flash/action_scripts/01_overview.html
@ALLA	http://www.allaire.com
@CF	http://www.macromedia.com/software/coldfusion/
@DEV	http://www.meta-magic.com/cgi-bin/fusewiki?DevNotes
@DIPO	http://www.digitalpoint.com
@DKAT	http://www.dkat.com/db
@FUSE	http://www.fusebox.org
@HIP	http://www.hipdesign.com
@JOST	http://www.johnstark.com/pd63.html
@MACR	http://www.macromedia.com
@MAG	http://www.magicsolutions.com/
@MONK	http://www.webmonkey.com
@ROSE	http://www.rational.com/products/rose/index.jsp

EHRENWÖRTLICHE ERKLÄRUNG

Hiermit erkläre ich, Steffen Nesper, geboren am 28. Juli 1975 in Mutlangen, ehrenwörtlich,

dass ich meine Diplomarbeit (thesis) mit dem Titel

„Procurement of a Client Management System (CMS) for Haaland Internet Productions“

unter Anleitung von Herrn Professor Dr. Dieter Schaal selbstständig und ohne fremde Hilfe angefertigt habe und keine anderen als in der Abhandlung angeführten Hilfen benutzt habe;

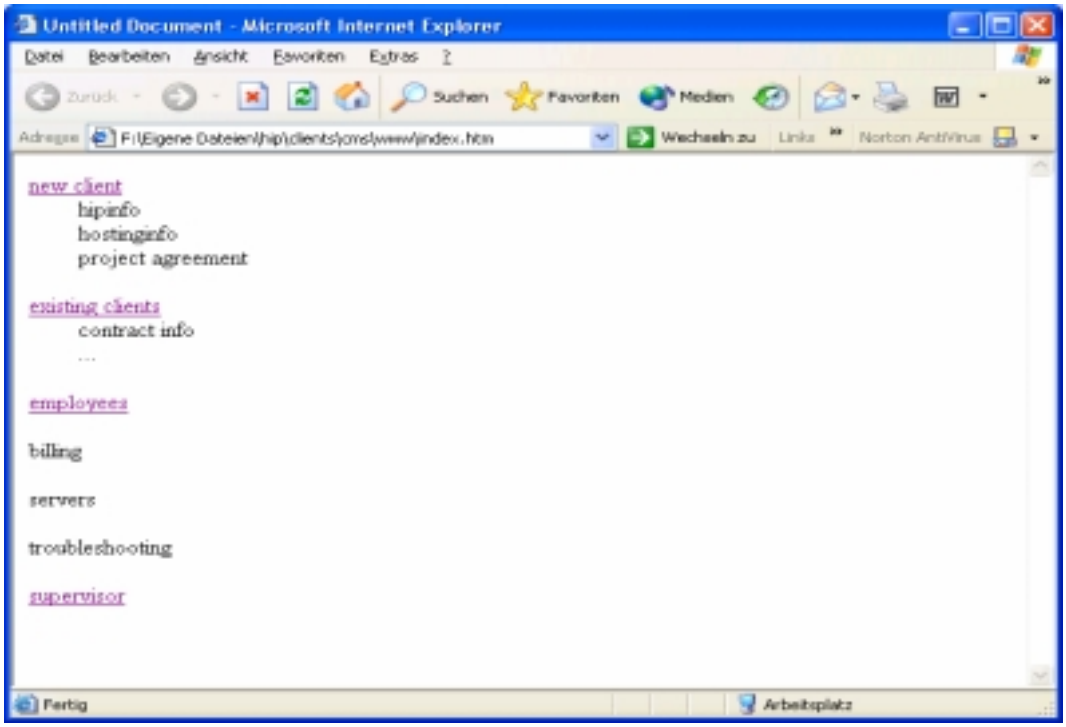
(2) dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

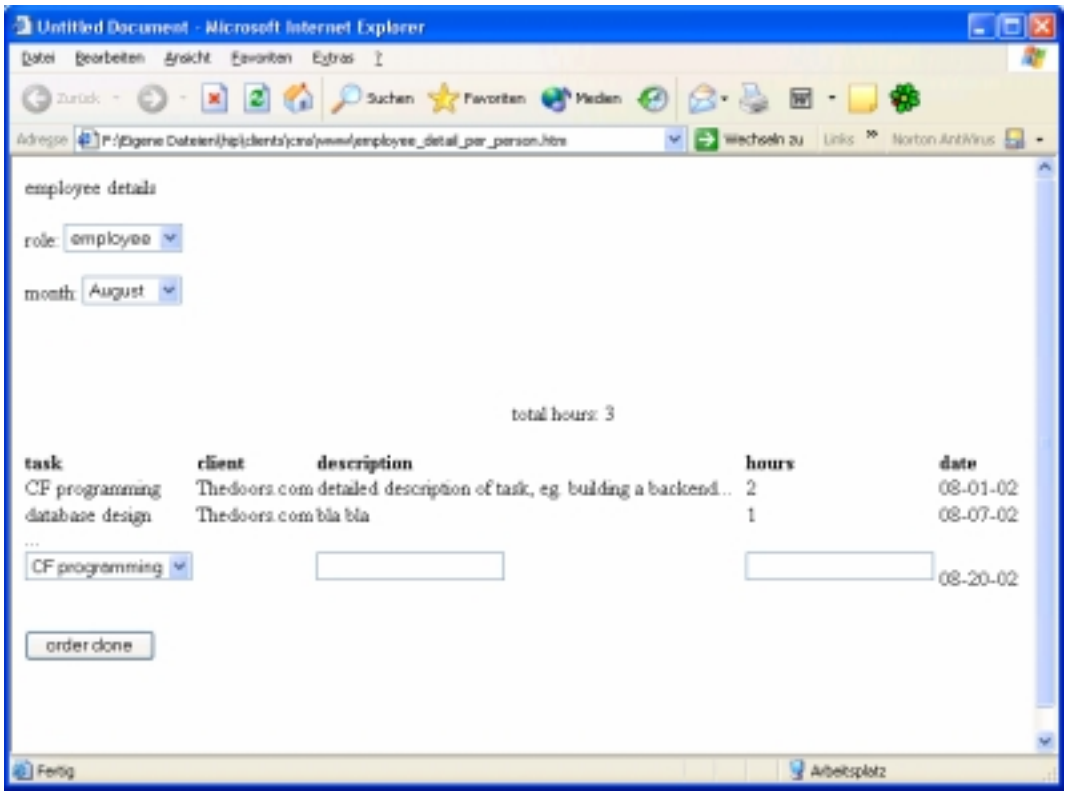
_____

Steffen Nesper

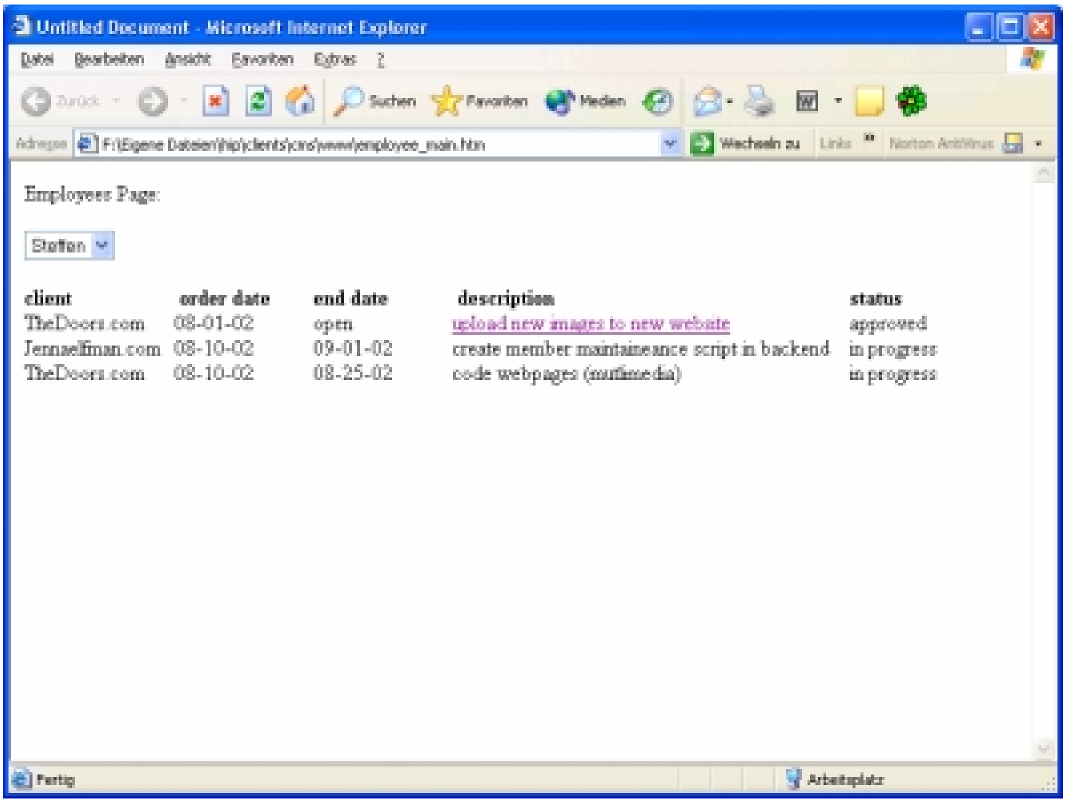
APPENDIXES



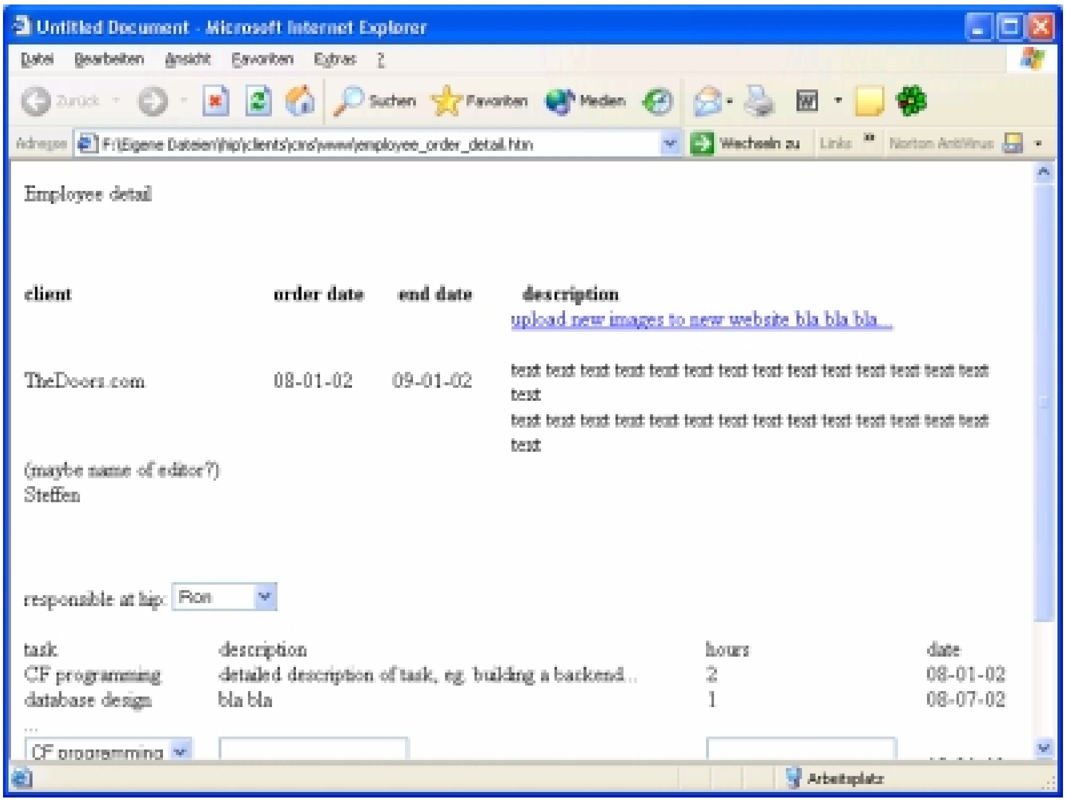
Appendix 1: Wire frame, home page



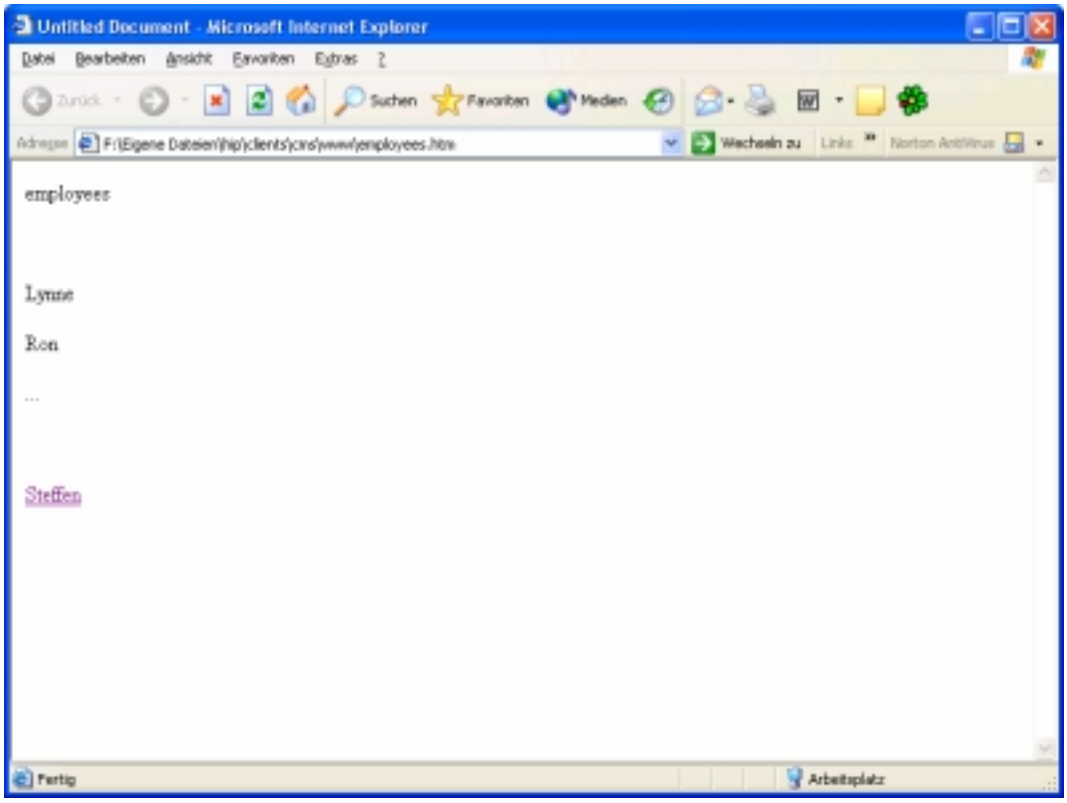
Appendix 2: Wire frame, employee details (overview)



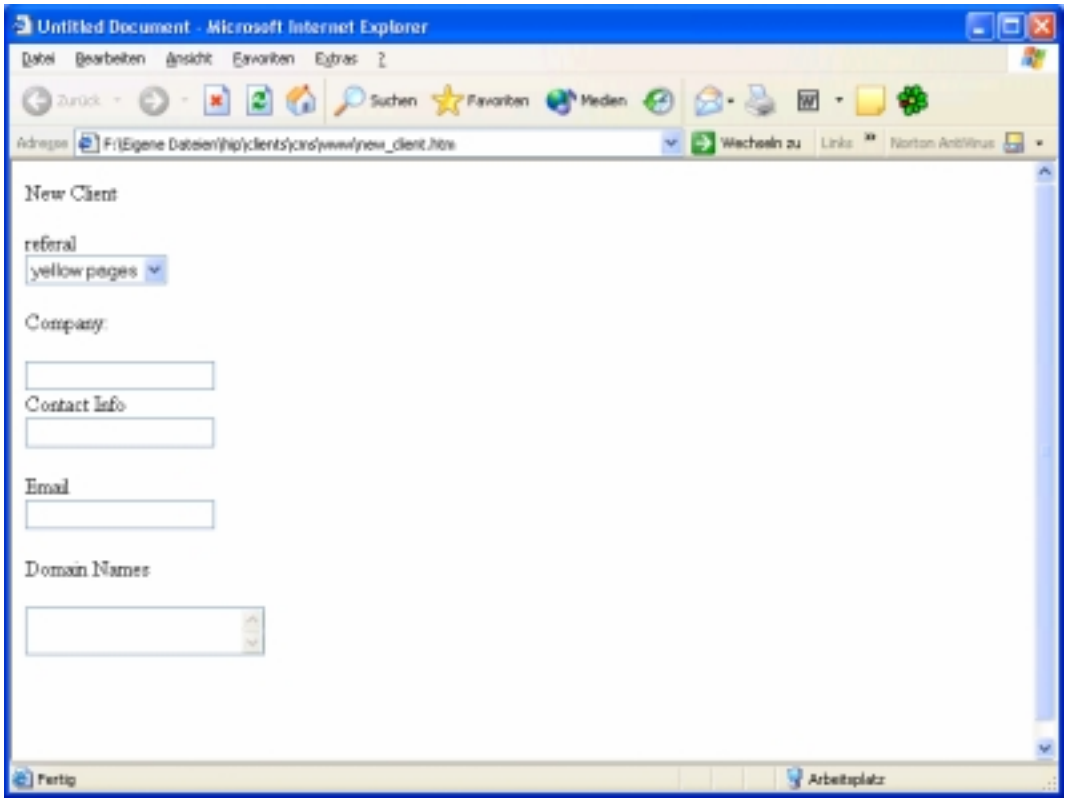
Appendix 3: Wire frame, task list



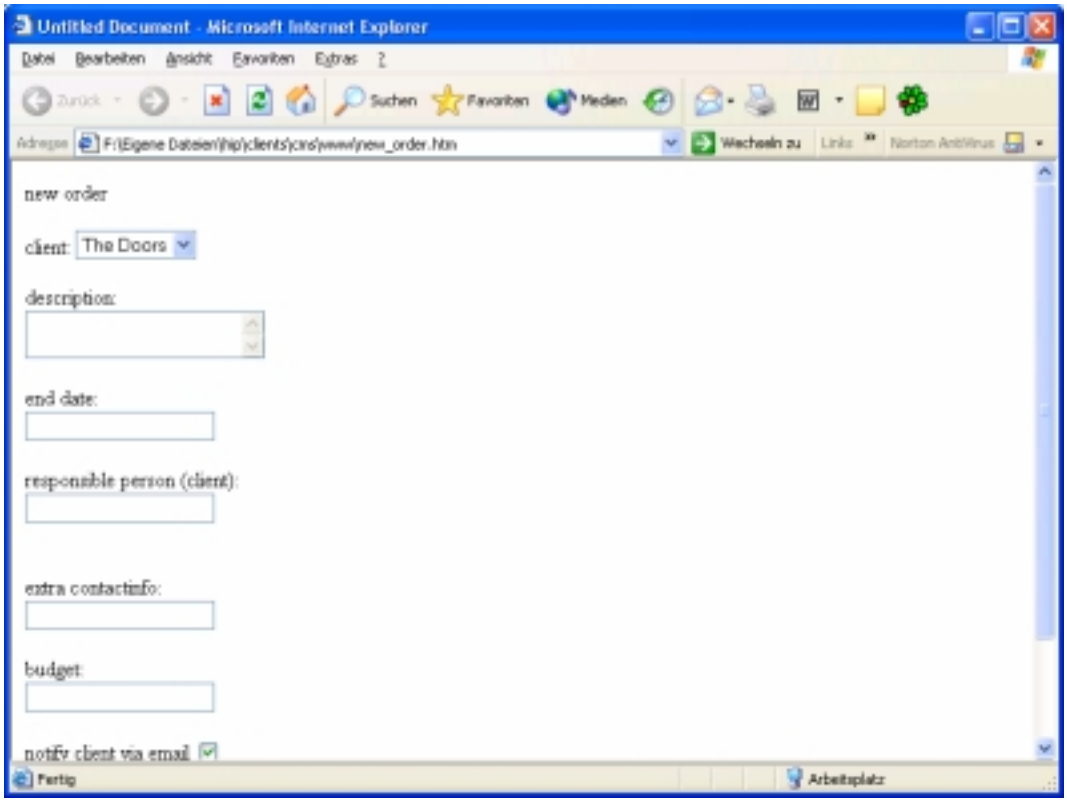
Appendix 4: Wire frame, employee detail



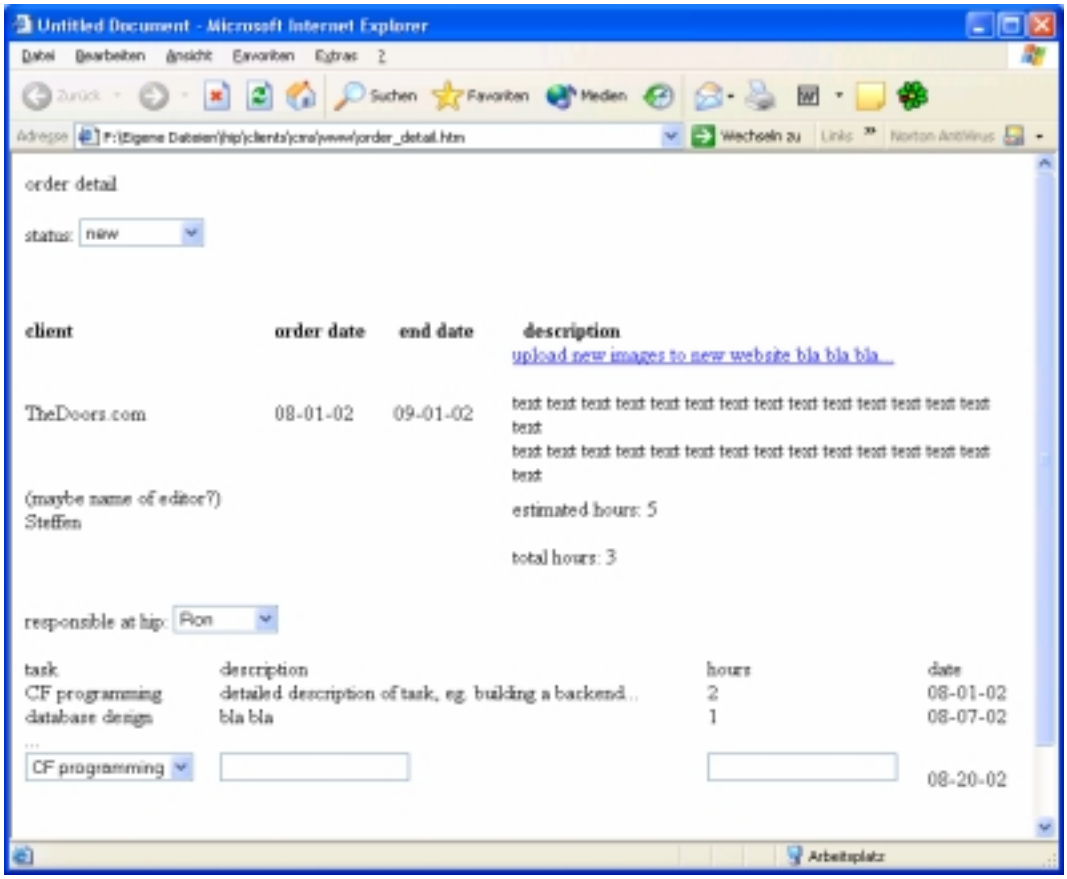
Appendix 5: Wire frame, employee list



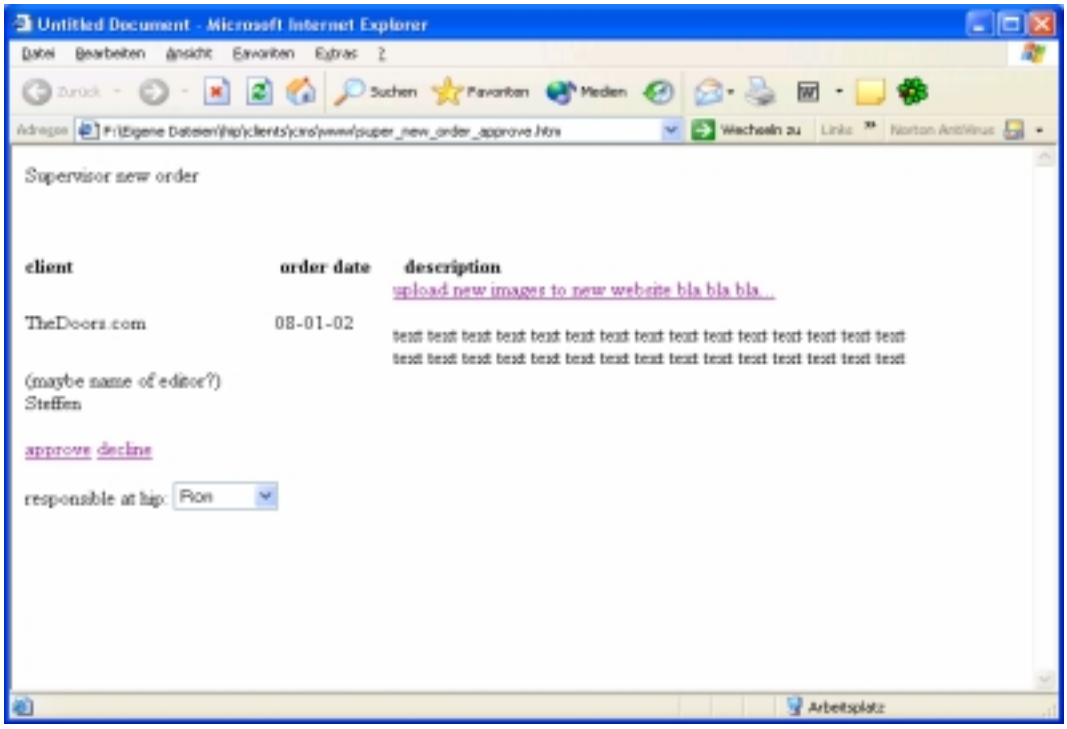
Appendix 6: Wire frame, new client



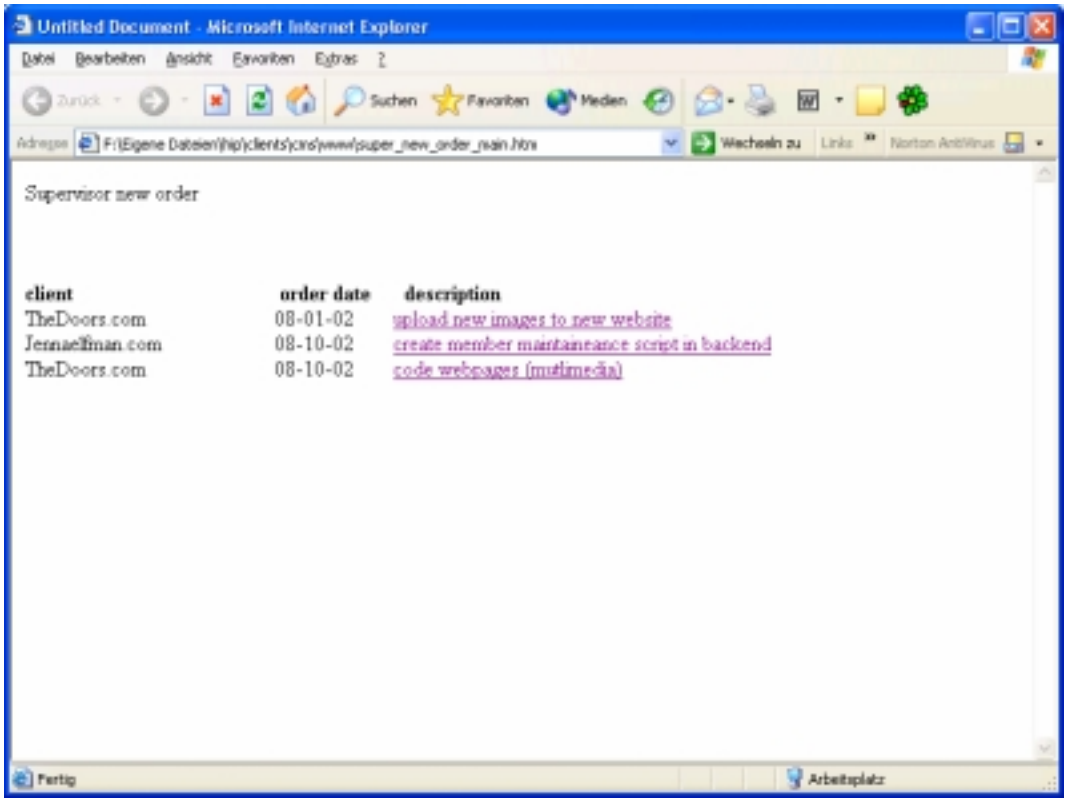
Appendix 7: Wire frame, new order



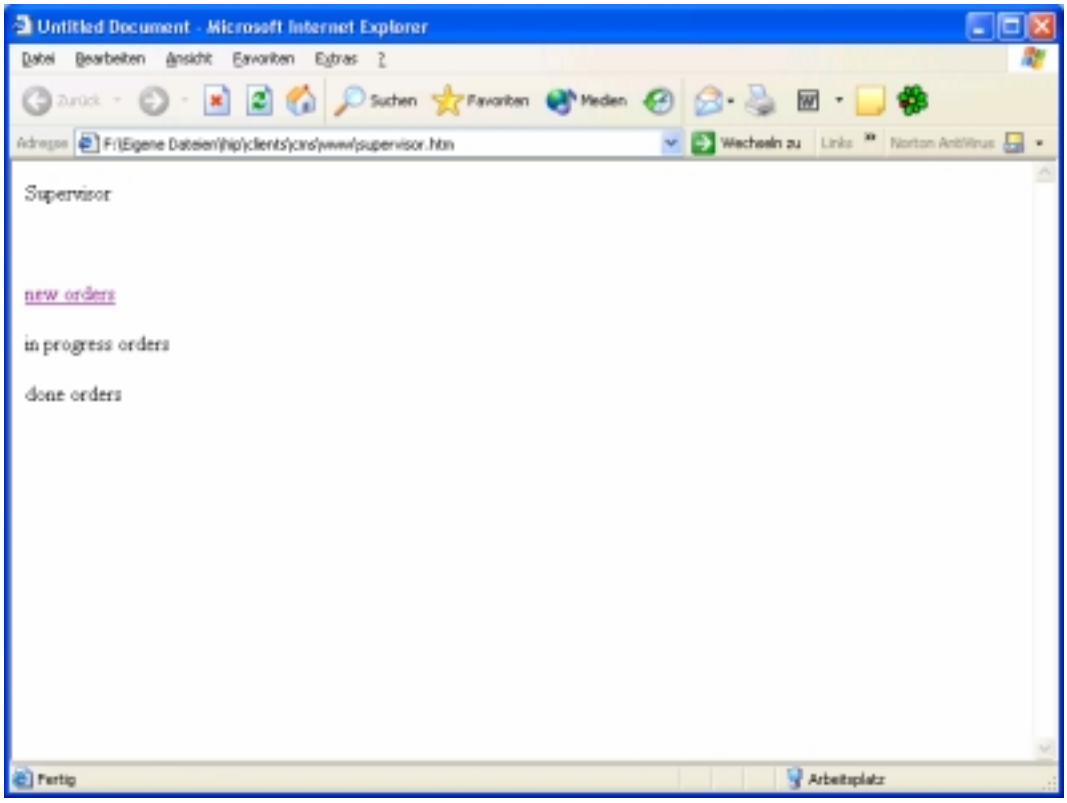
Appendix 8: Wire frame, order details



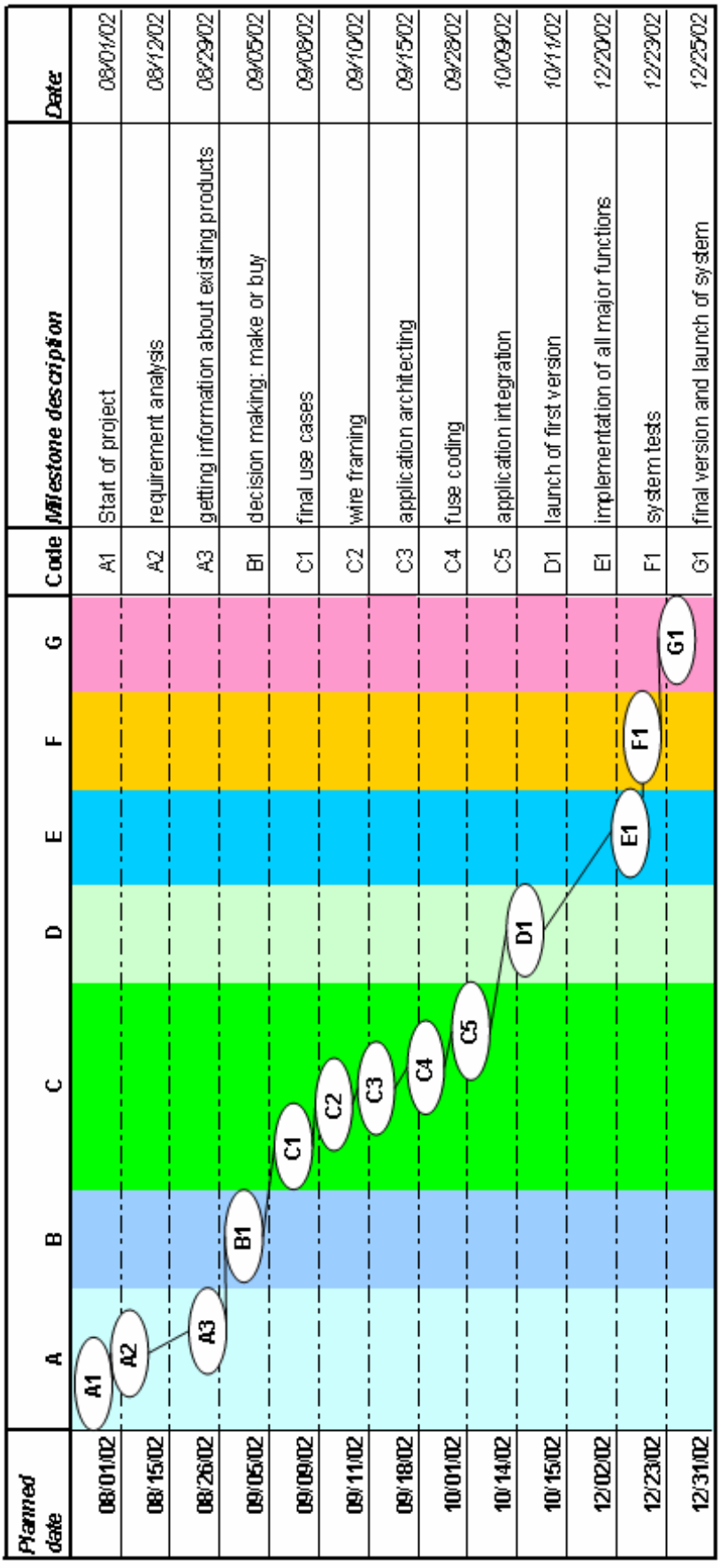
Appendix 9: Wire frame, supervisor order detail



Appendix 10: Wire frame, supervisor new order



Appendix 11: Wire frame, supervisor



Appendix 12: Milestone plan